NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

AN EFFECTIVENESS STUDY FOR PRIORITIZATION ALGORITHMS IN A COMMUNICATIONS NODE MODEL FOR THE COPERNICUS TACTICAL DATA INFORMATION EXCHANGE SYSTEM (TADIXS).

by

Christopher H. Halton

September 1997

Principal Advisor:	Michael G. Sovereign
Associate Advisor:	Orrin E. Marvel

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
September, 1997

3. REPORT TYPE AND DATES COVERED
Master's Thesis

4. TITLE AND SUBTITLE
AN EFFECTIVENESS STUDY FOR PRIORITIZATION ALGORITHMS IN A COMMUNICATIONS NODE MODEL FOR THE COPERNICUS TACTICAL DATA INFORMATION EXCHANGE SYSTEM (TADIXS).

5. FUNDING NUMBERS

6. AUTHOR(S)

Halton, Christopher H.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
The U.S. Navy has published its vision of the future in Command, Control, Communications, Computers, and Intelligence (C4I): Copernicus. Copernicus takes advantage of new technology and attempts to answer the demand for larger amounts of more timely information. Despite the advances in technology, new transmission methods and increased bandwidth, the U.S. Navy still does not have all the communications throughput that it desires. The author examines message prioritization algorithms as a way of making more efficient use of scarce communications resources. Through a simple communications node model and two algorithms, it is statistically proven that prioritization algorithms can improve the efficiency of a communication system.

14. SUBJECT TERMS
Copernicus, Tactical Data Information Exchange System (TADIXS), Algorithms, Prioritization.

15. NUMBER OF PAGES
142

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

AN EFFECTIVENESS STUDY FOR PRIORITIZATION ALGORITHMS IN A COMMUNICATIONS NODE MODEL FOR THE COPERNICUS TACTICAL DATA INFORMATION EXCHANGE SYSTEM (TADIXS).

Christopher H. Halton
Lieutenant Commander, United States Navy
B.S., University of Idaho, 1986

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY
(Joint Command, Control, and Communications)

from the

NAVAL POSTGRADUATE SCHOOL
September 1997

Author: Christopher H. Halton
Christopher H. Halton

Approved by: Michael G. Sovereign
Michael G. Sovereign, Primary Advisor

Orrin E. Marvel
Orrin E. Marvel, Associate Advisor

Dan C. Boger
Dan C. Boger, Chairman
Command, Control, and Communications Academic Group

ABSTRACT

The U.S. Navy has published its vision of the future in Command, Control, Communications, Computers and Intelligence (C4I): Copernicus. Copernicus takes advantage of new technology and attempts to answer the demand for larger amounts of more timely information. Despite the advances in technology, new transmission methods and increased bandwidth, the U.S. Navy still does not have all the communications throughput that it desires. The author examines message prioritization algorithms as a way of making more efficient use of scarce communications resources. Through a simple communication node model and two algorithms, it is statistically proven that prioritization algorithms can improve the efficiency of a communication system.

TABLE OF CONTENTS

I. INTRODUCTION

A. PURPOSE OF THESIS

The purpose of this thesis is to examine whether prioritization algorithms in a communication node have an effect on the measure of performance, in this case, average message delay before being transmitted from the node. Despite advances in technology, new transmission methods and increased bandwidth, the U.S. Navy does not have all the communications throughput that it desires. In an age of smaller budgets and increasing commercial demand for communications frequencies, the U.S. Navy must use its available resources more efficiently and effectively. Improving prioritization algorithms is one way of ensuring that the most important information arrives where it is needed first while less important inoformation still arrives.

B. BACKGROUND

In 1991, the U.S. Navy published its vision of the future in command, control, communications, computers and intelligence (C4I): Copernicus. Copernicus is a change in doctrine and technology that will allow the transmission of information in forms, methods and rates never before thought possible. In 1995, the Chief of Naval Operations reiterated that Copernicus is the Navy's C4I architecture for the future. This architecture will process all types of data and use many different transmission mediums. [Copernicus Forward, 1995, p. 1] Message processing and prioritization will be a vital part of this architecture.

C. METHODOLOGY

In this thesis, a communications node model is built and two different prioritization algorithms tested. Using measures of performance, the effect of the change in algorithms is statistically measured.

D. SUMMARY

Chapter II gives an overall description of Copernicus and in particular TADIXS (Tactical Data Information Exchange System), the main focus for this thesis in Copernicus. This gives the background for the reader to understand the basis for the thesis. Chapter III addresses the development of the model and the choices regarding the model and algorithms made during the development. Chapter IV describes the model and the two algorithms in more detail. Chapter V discusses the results of the model runs. The appendices contain the actual computer code and data analysis results.

II. COPERNICUS DESCRIPTION

A. COPERNICUS

"Naval Command and Control is the warfare function through which a maritime commander delegates his warfighting responsibilities to subordinate commanders and their units under his command."[Copernicus Architecture, 1991, p.1-1] Since the establishment of Space and Electronic Warfare (SEW) as a warfare area in 1989, command and control (C2), as encompassed inside the larger command, control, communications, computers and intelligence (C4I) technological and organizational system, has assumed new importance in the maritime warfare arena. The U.S. Navy has drastically revised its C4I doctrine and technology in response to leaps in technology, an increasing demand by warfighters for larger amounts of more timely information and the C2 issues raised by a growing battlespace. The result of that change in doctrine and technology is the system called Copernicus. Copernicus is the U.S. Navy's architectural and technological implementation of C4I for the 21st century. [Copernicus Architecture, 1991, p.1-1]

1. Concept

Copernicus, as an architecture, represents recognition of a world made smaller by increased technology, high data rate communications, long range sensors and weapons. The term Infosphere has been used to describe the high speed, seamless exchange of data on a global scale. Copernicus is meant to establish and take advantage of an "Infosphere" with the emphasis being on correct and timely information for the warfighter. The renewed emphasis on the warfighter has resulted in three new concepts: 1) User Pull, 2) Producer Push and 3) Virtual Circuits.

User pull is a new concept in which the user of information (the warfighter) will pull the information he needs from the infosphere. This concept, made conceivable by new technology, is the direct reaction to information overload at the warfighter level. With current communication systems, there is limited ability to separate critical sensor and operational traffic from mission support or administrative traffic. In addition, multiple source information as well as multiple routing schemes lead to repetitive information, overtaxed communication resources, interoperability, and security issues. [Rand, 1992, p. 5] User pull allows the warfighter to customize the information he receives and promises the prompt delivery of what the warfighter considers to be essential. User pull also includes the ability of the user to extract, upon demand, any information contained in the infosphere. [User Pull-White Paper, 1993, p. 1]

Producer push consists of information being generated at "producer" facilities, such as intelligence centers, which will be pushed to the warfigher (user) independent of any demand or request. So as not to maintain the status quo, the producer push is tailored toward the user's specific missions and will be adapted to changes in user's missions and status. [User Pull-White Paper, 1993, p. 2] The move from concept to technical definition is currently in progress for user pull and producer push.

The other major concept in the Copernicus architecture is the idea of virtual circuits. A virtual circuit (or network) is a circuit which is set up temporarily in order to allow for the efficient transfer of required data. The virtual circuit may consist of several different types of communication equipment to send the data. This idea is quite different from the U.S. Navy's current architecture of permanent, "hard wired" circuits. The entire life of a virtual circuit may range from 5 minutes to 5 hours to 5 days. [Copernicus Architecture, 1991, p. 3-1] The concept of virtual circuits has successfully

been tested in exercises such as Joint Warfighting Interoperability Demonstrations (JWIDs). Perhaps Copernicus' most important attribute is the implementation of the above concepts while defining and forming this architecture.

2. Implementation

Copernicus provides an architecture, using the new concepts mentioned above, which will lead to a technological implementation. Initially, the Copernicus architecture consisted of four pillars: the Global Information Exchange Systems (GLOBIXS), the CINC Command Complex (CCC), the Tactical Data Information Exchange Systems (TADIXS), and the Tactical Command Center (TCC). By 1995, Copernicus had evolved into five pillars. The fifth pillar, Battlecube Information Exchange System (BCIXS), extends the architecture to include the battlecube, the area in which shooters and weapons reside and are used.[Copernicus Forward, 1995, p.5] With these five pillars, the Copernicus architecture will act as an interactive framework that ties together the C2 process of the Joint Task Force (JTF) commander, the Navy tactical commander afloat, the numbered fleet commander and others with the CINC's ashore. [Copernicus Architecture, 1991, p. 3-1]

GLOBIXS, the first pillar, are virtual networks that link the command and activities ashore to support the forces afloat. They are configured on a theater or worldwide basis and are constructed to transport, standardize, and concentrate shore-based sensor, analytic, command support, administrative, and other data for further passage to commanders afloat. GLOBIXS will be constructed like interstate highways--they are limited-access, high speed and highly concentrated. In addition, they have connections among each other so that traffic may be shunted across several GLOBIXS as well as to the operating forces through a consolidated CINC Command Complex (CCC). [Copernicus Architecture, 1991, p. 4-1]

The number and nature of GLOBIXS is intended to be dynamic, so the architecture can support future command structures and individual CINC unique priorities. There are to be eight standing GLOBIXS around the world. They are the following: SIGINT GLOBIXS, Anti-Submarine Warfare (ASW) GLOBIXS, SEW GLOBIXS, Imagery GLOBIXS, Data Base Management GLOBIXS, Command GLOBIXS (a multi-media net connecting CINC's, JTF Commanders, numbered fleet commanders, etc.), Research and Development Information Exchange System, and Navy Information Exchange System (NAVIXS). NAVIXS will be the Navy implementation of the Defense Message System. The GLOBIXS will use current and future common-user communication systems, such as the Defense Communication System, as vehicles for network communications. [Copernicus Architecture, 1991, pp. 4-1,2]

The CINC Command Complex (CCC) is the second pillar of Copernicus. The CCC will include a number of existing organizations brought together technologically by common workstations connected to a metropolitan area network (MAN). Like the GLOBIXS, the CCC is a virtual network. The CCC MAN will provide the "information highway" over which GLOBIXS and Tactical Data Information Exchange System (TADIXS) data will travel, as well as that data generated at the CCC. [Copernicus Architecture, 1991, p. 5-1]

The GLOBIXS will terminate into the CCC. In addition, the CCC MAN will be connected to many local area networks (LANs) contained within the organizations that collectively make up the CCC. Because the CCC includes a MAN, the CCC should be viewed as an extremely flexible construct that could include Navy and Non-Navy agencies and organizations as required by the CINC. One should keep in mind that the GLOBIXS is an aggregation of

"communities of common interest" while the CCC is aggregation of CINC command structures ashore. [Copernicus Architecture, 1991, p. 5-1]

There are six organizational building blocks envisioned to comprise the core of a CCC. They are the following: Fleet Command Center, Operations Watch Center (a collection of GLOBIXS anchor desks acting as a gateway for the at sea Composite Warfare Commander), the SEW Center, the Research Center, the Joint Intelligence Center, and finally the ASW Center. These centers, in the aggregate a CCC, will serve as the centralized C4I center for the implementation of the missions assigned to the CINC. The CCC supports the commander by processing, displaying, and disseminating organic and non-organic information to provide a clear picture of operations within the theater. This information is the basis for plans of action and force direction decisions. [Copernicus Architecture, 1991, pp. 5-4,4]

As part of the CCC duties, the CCC personnel will anchor--filter, sort, analyze and move--GLOBIXS information for the tactical commander. The GLOBIXS will afford the Composite Warfare Commander (CWC) the capability to receive the information tailored to his needs in order to fulfill his specific mission. If he so desires, the CWC may decide that some or all GLOBIXS information be anchored by afloat personnel, based upon his personal preference. With full implementation of Copernicus, the CCC anchors will act as interfaces, or gateways, between the GLOBIXS and TADIXS virtual networks. Information taken from their respective GLOBIXS networks will be filtered and consolidated into a concise, uniform package that can be sent over the TADIXS to the Tactical Command Centers (TCC) discussed later. These same personnel will similarly transmit "anchored" TADIXS information over their respective GLOBIXS networks. [Rand, 1992, p. 8-9]

The TADIXS, the third pillar, will be the link that provides a shared, common tactical picture in the CCC and the TCC. [Copernicus Architecture, 1991, p. 6-1] TADIXS are to be the virtual networks which will support afloat TCCs. TADIXS are envisioned as information nets time-sharing communication circuitry over a broad range of bearer services, transmission media such as UHF, SHF, EHF, commercial SATCOM and HF. The information of one TADIXS may be supported by several channels and, conversely, one channel may support several TADIXS. [Rand, 1992, p. 16] This is also known as dynamic resource allocation.

The number of TADIXS will not be fixed; instead, they will be connected for the length of time necessary to transport the data to the subscribers and then broken. Because of this, TADIXS have been grouped into four broad categories, somewhat analogous to GLOBIXS. The four categories are Command TADIXS, Support TADIXS, Direct Targeting TADIXS, and Force Operations TADIXS. [Copernicus Architecture, 1991, pp. 6-1,2] Initially, the actual implementation of TADIXS was to be the Communication Support System (CSS). CSS has been replaced by the Joint Maritime Communications Strategy (JMCOMS). JMCOMS implements the tactical communications segment of the Copernicus C4I architecture. JMCOMS has a three pronged approach: Automated Digital Network System (ADNS), SLICE Strategy (implemented as Digital Modular Radio) and the Integrated Terminal Program (ITP). [JMCOMS Overview, 1997, p.1]

The fourth pillar in the Copernicus architecture is the Tactical Command Center (TCC). The TCC is intended to signify the combat "nerve centers" of the tactical commander and his units. Thus, the TCC in Copernicus means not only the Tactical Flag Command Center (TFCC), Combat Information Center (CIC), and other C4I spaces/centers on a flagship, but also the tactical centers for individual units. Architecturally,

the TCC is analogous to the CCC. The TCC provides the tactical displays, integrated information management, and accessibility to tactical communications to support Navy warfighting missions. Both the CCC and the TCC will share a consistent tactical picture and connect the Navy to the Services and to allies, at the tactical level and the theater level. With the establishment of fiber optic busses afloat, the LAN connectivity used in the TCC will become virtual and allow for high speed, high bandwidth data transmission.

The final pillar in the Copernicus architecture is the Battlecube Information Exchange System (BCIXS). The battlecube is a conceptual, multi-dimensional area that includes subsurface, surface, air and space as the environment for conducting warfare. BCIXS represents the battlecube in which tactical forces operate. BCIXS boundaries are fluid and defined by the dynamics of the battle. Shooters operating in the battlecube form the operational nodes in the BCIXS. Shooters are equipped with C4I tools that allow them to receive and process information from the Copernicus architecture.

B. JOINT MARITIME COMMUNICATIONS STRATEGY

As mentioned earlier, the initial implementation of the TADIXS pillar of Copernicus was to be the Communication Support System (CSS). To support the gradual implementation of the Copernicus communications segment, a new technical and program strategy, replacing CSS, has been implemented called the Joint Maritime Communications Strategy (JMCOMS). JMCOMS incorporates the latest advances in commercial and military communications technology to maximize bandwidth, enabling the sharing of information seamlessly, in real- or near real-time, through flexible, adaptive and interoperable systems and services. "JMCOMS' rapid, reliable, and

reconfigurable communications connectivity to all echelons of command and its accompanying information transfer infrastructure make the sensor-to-shooter construct a reality in the C4I environment." [JMCOMS Overview, 1997, p.1]

To accomplish its mission, JMCOMS has taken a three pronged approach: Automated Digital Network System (ADNS), the SLICE strategy (Digital Modular Radio (DMR)) and Integrated Terminal Program (ITP). ADNS forms the backbone of JMCOMS. ADNS uses off the shelf protocols, processors and routers to create a robust and flexible networking environment. Currently, Internet Protocols, Asynchronous Transfer Mode and other commercial products are being adopted or adapted. Interfaces to all RF media from HF to EHF provide the total throughput and access needed. Networking techniques make efficient use of available channels. [JMCOMS Overview, 1997, p.3]

The SLICE strategy uses digital implementation of new modulation techniques, coding strategies and encryption devices. Powerful signal processing and software reconfigurable radios will be implemented in a compact, economical bus environment resulting in more radio for less money. DMR, which implements the SLICE strategy, covers <2 Ghz terrestrial and SATCOM (satellite communications) requirements. The Integrated Terminal Program (ITP) is a strategy to meet future requirements for high capacity satellite communications for ships, submarines and shore commands in a cost-effective manner. ITP intends to migrate current SATCOM systems which operate above 2 Ghz to open architecture, modular, multi-band terminals and low observable antennas.

C. DEVELOPMENT OF THESIS

The Navy's current message prioritization scheme has only four different sender assigned precedences. From lowest priority to highest priority the categories are Routine, Priority, Immediate and Flash. Each precedence has a delivery time limit associated with it and a higher priority message (i.e. Flash vs. Priority) will automatically supersede the lower priority message and be sent first. If there is a queue for message traffic, then the queue operates on a First In, First Out (FIFO) basis within each precedence. When the message system is heavily loaded, large delays result for the lower priority messages. In fact, during Desert Shield/Desert Storm (DS/DS), there were delays of two to three days for Immediate messages, which are normally required to be delivered within 5 minutes of transmission. These long delays cause a lack of faith in operators in the message system to get messages through in a timely manner. It is surmised that messages are eventually given higher priorities than normally required to help ensure timely delivery. In DS/DS, it was also found that numerous messages and information were sent out repetitively to help ensure receipt. Combined with large numbers of messages that contain information unimportant to the warfighter, information overload occurs at the area least able to handle or afford it--the warfighter.

To help combat this information overload, the Navy has developed the new C4I concept named Copernicus. Copernicus, as mentioned earlier, has a new way of dealing with communications, dynamic resource allocation, as well as a new way of looking at information, user pull or producer push. Currently, these new ideas are still being researched, explored as well as defined. Some of the problems inside these new areas deal with prioritization of messages and information. What type of information is user pull (or can we expect to be pulled by the user) vice producer push? Is user pull given a higher priority than producer push simply because the user (warfighter) asked for

the information? For system development purposes, what ratio can be expected for producer push information to user pull information? While this thesis does not address these questions, it will look at prioritization and its effect on system throughput.

This thesis looks at prioritization under the new concept of Copernicus and ADNS. With a simple communications node model and different algorithms, this thesis looks at whether or not prioritization/ reprioritization algorithms would increase efficient use of limited communications assets. The result of these prioritization algorithms may mean that if all users of the message/information system know that all messages will get through in a "reasonable" amount of time, trust in the systems will go up and message precedence inflation need not occur. If the delivery time for Flash messages was slightly longer, what would that do to the overall average delivery time as well as for each precedence? This thesis, using operational analysis modeling and statistical techniques, gives some examples as to what may determine the priority of a message vice the current precedence system. In addition, the thesis uses user pull as a feature in the prioritization algorithms in order to see what the effect might be in the Navy's developing and future message handling/C4I systems. This is important since one of the major emphases of Copernicus is user pull. With this radical change from current communications systems, the effect of user pull should not be overlooked in terms of its effect on the Navy's message handling/C4I systems.

D. CHAPTER SUMMARY

This chapter has given an overview of the U.S. Navy's new C4I concept Copernicus, described the communications segment of Copernicus. The next chapter will describe how the communications node model was developed.

III. COMMUNICATION NODE MODEL DEVELOPMENT

This chapter lays out the foundation for the development of the communication node model and the algorithms. The model developed is a simple one but provides an effective way of simulating and measuring the effect of different prioritization algorithms. The limitations of the model are discussed in this chapter as well as the assumptions for the model. Some aspects of job shop scheduling were used in the algorithms that were tested against the base case and so a brief review of job shop scheduling is also included.

A. MODEL GOALS

As the Navy changes the way it communicates information, thought may be given to changing some of the classifications or criteria used for data/information. Additionally, in these days of fiscal austerity, is it possible to more efficiently utilize the Navy's communication assets by making software vice hardware changes. That is the crux of this thesis. By changing how the Navy prioritizes its messages and information, messages of all precedence types will have shorter average delivery times.

As mentioned earlier in this thesis, the current or base case of prioritization of messages in the Navy is the sender assigning a precedence. Then the communication node places the message in a FIFO queue with others of its precedence. Suppose a Routine precedence message arrives at a communications node at time zero. At time five, when an Immediate precedence message arrives at the communications node, even if the Routine message has not yet been sent out, the Immediate message will automatically be sent out before the Routine message. In fact, all of the higher precedence messages will be sent first. Thus during heavy loading, the lower precedence messages may not get sent out until loading drops substantially.

This modified FIFO queue algorithm of prioritization is what forms the base case in the model developed for this thesis. The results of the base case algorithm's performance under a heavily loaded message system are compared to the results of one other algorithm under similar loading. Also, the thesis looks at the potential effects that user pull and producer push might have on a message system. This thesis uses the model and associated algorithms to prove or disprove the concept that new prioritization algorithms may allow for more efficient use of scarce communication resources.

The base case algorithm is tested against another algorithm. The alternate algorithm takes into account several other characteristics of the messages before assigning a prioritization. The factors considered are the following: assigned precedence, information type contained in the message, length of the message, whether the information is user pull or producer push information, and finally, length of time in the queue. The alternate algorithm reprioritizes the queue at designated intervals in order to take into account the length of time a message has been waiting. This, of course, is a major difference between the base case and the alternate algorithm. The interval was chosen by the author. The model and algorithms will be discussed in detail later in the thesis.

B. JOB SHOP SCHEDULING

Job shop scheduling is an idea that comes out of the manufacturing business and the operations analysis world. A job shop is characterized by sets of equipment that are used in the manufacture of different and diverse orders. The sequence of the orders or jobs through the sets of equipment may differ substantially; thus causing scheduling and flow problems. [Groff, 1972, p. 437] Job shop scheduling seeks to minimize the average flow time

through the job shop, minimize the average waiting time and minimize average lateness. These in turn have an effect on the average utilization of the shop. [Groff, 1972, p. 439] This appears to be similar to a message system in which different messages (or data) must use the same equipment, but with different routings, etc. This appears to be exactly what a system like ADNS will address at a communications node.

Three factors are often considered when doing job shop scheduling. First, include a function of the job due date to pace the progress of individual jobs and reduce the variance of the lateness distribution. Second, include some consideration of the job processing time to reduce congestion and to get jobs through the shop as quickly as possible. Third, include some foresight to avoid selecting a job from a queue which, when the current operation is completed, will move on to another queue which is already congested. [Groff, 1972, p. 442] Another factor which is not usually mentioned is one of management priorities. Jobs which management views as being the most important also require extra consideration in the scheduling decision, just like the assignment of a priority to a message by the sender. These four criteria match up well to the qualities of a message handling system. One factor that is not really applicable is the factor regarding loading at the next operation for a job. Especially with virtual circuits and networks, when a message is sent, it is expected that the system has the capacity to route the message all the way to its destination. The processing time of a manufacturing job matches well with the processing time of a message (size of a message divided by the data rate of the system). The due date of a manufacturing job also matches well with the precedence concept, in that the precedence, with the required delivery time associated with each precedence, indicates a desired "no later than" delivery time.

The goals of job shop scheduling also match well with a message handling system. Communications managers desire to reduce the average delay in sending a message for all messages. In addition, increasing the degree of utilization of limited communication assets is always a goal. Reducing the lateness of messages is certainly a worthy goal.

C. THE MODEL VERSUS REAL WORLD

As ADNS is actually implemented, it is expected to prioritize all message and information traffic. At this point, however, ADNS does not exist in its final form. Due to this, many assumptions were made and a simple communications node model was designed. This also reflects an emphasis on algorithm development and coding. When ADNS reaches its final form, the system will be a very complicated, software intensive system. For the purposes of proof of concept, however, the model developed, shown in Figure 1 below, is believed to be sufficient to illustrate the concept.

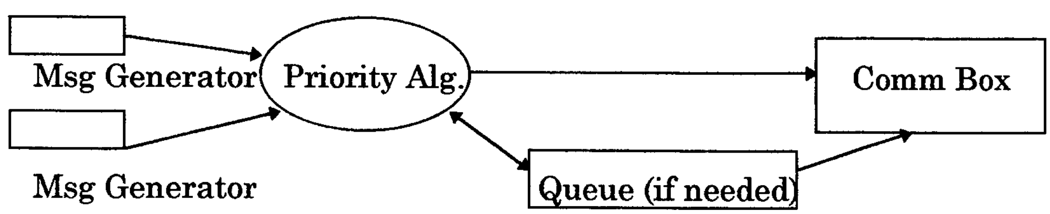

Figure 1. Model visual depiction.

The model consists of two message generators, each generating messages independently, an algorithm to prioritize the generated messages as well as reprioritize the queue, plus one communication box. The communication box plays the role of the transmission media. When free, it immediately sends a message. If transmitting a message, it waits until it is

free and then pulls the next message from the queue. The model will be discussed in more detail at a later point in the thesis.

D. MODEL ASSUMPTIONS

The following are the assumptions associated with the communications node model. Because the emphasis of this thesis is the effect of prioritization on the communications node throughput, many assumptions were made to keep the model simple. The effect of prioritization algorithms on the communications node throughput should be the same for a simple or complex model given that all other elements are held equal.

- Assumed the message generators' data rate as well as the pulling or outgoing data rate.
- Nearly fully loaded system. The system was assumed to be close to but not exceeding full capacity. This was to avoid complications with an overloaded system, i.e. one that will rarely be capable of sending all messages. The loading of the system was at levels of 80, 90 and 95%.
- Message sizes. With no data easily available regarding message size distribution, a uniform random distribution was assumed.
- Information types. Fourteen different information types were assumed, in order to have a broad base without being overwhelming. The information type breakdown is as follows: normal peacetime operations, intelligence reports, ship and troop movements, weather, aircraft movements, reports of enemy contact, reports of unusual major movements of military forces in peace or strained relations, enemy counter attack, request for or cancellation of additional support, widespread civil disorder/grave national disaster, distress assistance, operational plans concerning projected operations, major strategic decisions, administrative, logistic and personnel matters.
- User pull information. It was assumed that 25% of messages in each information type from message generator 1 were user pull data. For message generator 2, it was assumed that the ratio of user pull to producer was different and so 75 to 25 was assumed. In addition, it was assumed that user pull information should be given a higher priority than pushed information.

- Distribution of information types. There was a uniform random distribution for each message type assumed. In other words, no information type was more likely than another.
- Distribution of precedence type. A uniform random distribution of the four precedence types was used, e.g., 25% each. This also reflected the high stressed, high loading of the message system in that there were much more high precedence messages than are normal in peacetime operations.

With the above assumptions, the model and algorithms were considered detailed and accurate enough to allow for the concept to be examined. Again, the focus of this thesis is the effect of the algorithms on virtually any communication node.

E. CODE CHOICE

The author looked at several different COTS (commercial off the shelf) computer software codes to see which would be suitable for the purposes of this model. CommNet and OpNet were, at first, the most favored candidates since they are network and communications modeling and simulation software. However, each package of software required detailed information regarding the communication system. As previously noted, such details (i.e. packet overhead size) are not yet known. The software packages were also not located on the computers used by students for thesis and classwork. In addition, both software packages require modification for this study with computer languages with which the author has no programming experience.

Borland's Turbo Pascal was chosen because of the author's experience with the software. In addition to the author's experience with the software, Turbo Pascal has a relatively benign troubleshooting environment to allow for debugging. This troubleshooting environment was used quite extensively during the model and algorithm code development. With the selection of Turbo Pascal, several drawbacks were accepted. First, the language is not

designed for nor well suited for communication modeling. Second, more actual code would need to be written since no communication models existed in the software (vice Opnet, etc.).

Several lessons were learned while developing this thesis. One, Turbo Pascal limits the size of the data structures, thereby preventing one of the algorithms from being coded and tested. Two, because of Turbo Pascal's nature, the language is very difficult to use for a communication model. Three, the troubleshooting environment was very useful and very helpful in solving numerous code problems.

F. CHAPTER SUMMARY

This chapter has examined the communications node model development. It covered the goals of the model; how it resembles job shop scheduling and can therefore use many of the same measures of performance. In addition, the assumptions used in the model were stated as well as how it compares to how ADNS is expected to look and why Turbo Pascal was used. In the next chapter, the thesis will describe in detail the model and the algorithms.

IV. COMMUNICATIONS NODE MODEL DESCRIPTION

This chapter will discuss in detail the model developed for this thesis and the algorithms used to prioritize the messages generated in the model.

A. MODEL

The model, an event step simulation, consists of two message generators, an algorithm to prioritize the messages being generated, and a "comm box" which pulls the messages out of the system, analogous to the messages actually being transmitted from the platform. The message generators act independently, and provide messages with assumed Poisson distribution arrival times. The arrival times are generated using the Turbo Pascal uniform distribution random number generator and a formula which converts the random number into an arrival time. The average number of arrivals per minute varied between each set of runs.

When it was time for a message to be generated, the attributes of the messages were then generated using the random number generator: the message generators used a uniform distribution for the size of the message, the precedence of the message, and the information type of the message. There was a difference between the message generators for the distribution of user pull and producer push. Message generator 1 randomly generated user pull messages 25% of the time and producer push the other 75% of the time. Message generator 2 randomly generated producer push messages 75% of the time and user pull messages 25% of the time. The generation of the user pull or producer push message was independent of what the previous message had been, as was true for all the message attributes. The combined arrival rate of the two message generators, as measured in average message size (in bytes) multiplied by the average number of messages arriving each

minute, was set at .8, .9 and .95 of what the model was capable of pulling out of the model. This ensured that the model was fully loaded but not overloaded to a point that the model could not send out virtually all the messages.

The comm box pulls messages based upon a given data rate. That data rate is set at the beginning of a set of model runs and is measured in bytes per minute. The comm box pulls a message at one of two conditions. One, the queue is empty, the comm box is inactive, and a new message is generated. Therefore, the new message is sent straight to the comm box and is processed directly out of the model. Two, there are messages in the queue, the comm box finishes with the message it is currently sending out, and then pulls the next message from the front of the queue. If there is a queue, a message may not bypass the queue. The message must go into the queue and is then pulled in order of priority. One assumption of the comm box is that there is no down time for the comm box. It assumed that the message system works perfectly outside of the model, and the model can always send out a message when it is time to send.

B. PRIORITIZATION ALGORITHMS

The first algorithm developed was the base case algorithm, the FIFO queue. The code is contained in Appendix A. An array of four records was developed, one record for each precedence of message. Each record has a pointer to a linked list of all the messages in the queue of that precedence. Each record also has an integer with the number of messages in that particular linked list. If the comm box is busy sending out a message, then the newly generated message enters the queue. Each message goes into the linked list for its precedence and is pulled from that linked list in the order in which it arrived, first in, first out. When pulling messages from the queue

for the comm box, the algorithm takes all messages from the highest precedence first. Therefore, a Flash message will be sent first, no matter what other precedence messages may have arrived at the queue first, and so on through the precedences. This method was coded and successfully run on the test platform.

The second algorithm developed was the array of linked lists algorithm. The number of records in the array depends upon the maximum possible size of the messages being generated during that run. This is due to the size being a factor in the priority ranking of a message. Typically, the array was the maximum possible size of the message plus two hundred. This allows for the other attributes to be factored in to the ranking. Each record is for one priority ranking and includes a pointer to the linked list of messages in that priority ranking as well as the addresses of the records that have messages that are immediately above and below the priority ranking in question. When a message is put into the queue, a point value or priority ranking (a scalar value) is generated based upon the precedence, information type, size and user pull/producer push classification. The equation was Priority = Size + InfoType + Precedence with an additional thirty places removed from the priority ranking if the message was a user pull message. The lower the point value generated, the higher the priority of that particular message. At designated intervals, all the messages in the queue are pulled out of the queue and have a new priority ranking generated, this time including the length of time in the queue. The longer a message is in the queue, the higher its priority is, given all other attributes being constant. This algorithm was coded and successfully run on the test platform. The code is contained in Appendix B. Run attributes such as arrival times, maximum message size, etc., were kept small enough to avoid data structures that were too large for the test platform.

The final algorithm developed was the five dimension matrix with a search pattern. Each dimension of the matrix is a scale of one of the attributes of a message: precedence, information type, size, user pull/producer push, and time in the queue. Each message is placed in the matrix based upon its attributes. With this method, there is no explicit ranking of the worth of all attributes relative to each other. In addition, at designated intervals, all the messages in the matrix are pulled out of the matrix and reassigned a new priority. This reflects the time spent in the matrix (queue). The search pattern starts searching for a message in the "corner" of the matrix that consists of the smallest size, flash message precedence, maximum time in the queue, most important information type and user pull. The search pattern then searches through the matrix until it finds a message to send. At this point, the comm box sends the message and the search pattern, as long as there is at least one more message in the matrix, continues its search for the next message to be sent. Should a new message arrive in the part of the matrix that has already been searched, the search pattern will return to that message in preparation to send that message next. The search pattern developed for this algorithm would look at and send all the Flash messages first and then the information types that were considered the most important, the top four information types. From then on, the search pattern would increment an attribute, look to see if there was a message and then either send the message to the comm box or if no message, increment the next attribute. This algorithm was found to have data structures that were too large for the test platform and so was not coded or able to be tested in this thesis but may be interesting for follow on work.

C. MODEL AND ALGORITHM DRAWBACKS

This model was developed without much input data from "real world" systems. As such, the model does have significant drawbacks. These drawbacks may include incorrect distributions used for the random number generator, as well as for the percentage of user pull and producer push messages produced. The model has not been verified nor validated by any communication expert and so lacks those critical qualities. The model, however, does serve its purpose in allowing for the examination of the qualities of the different algorithms.

The algorithms are very different from one another and each has its disadvantages. The base case does not allow for any other factors to used in determining which messages should be sent out first. In the event of system overloading, the lowest precedence messages may take days instead of hours, Ala. Desert Shield/Desert Storm. In addition, the warfighter or communications person may not change or customize the prioritization algorithm to set his command needs or preferences.

The array of linked lists algorithm also has several drawbacks. When reducing the five attributes down to one scalar value, there is a weighting done of the attributes in relation to one another. While in this case all the attributes were weighted the same, in reality, the size plays the dominant role in determining priority. This is due to the large sizes the messages can take on in relation to the other attributes. As an example, two messages with similar attributes except size are being prioritized. The first message has a size of 1500 bytes and the second has a size of 500 bytes. With all other attributes held constant, the second message is 1000 points lower in the priority ranking and therefore will be sent much sooner than the first message. In addition, the first message could change every other attribute to its most important classification and still not approach the priority ranking of the second message. At this point, the array of linked lists algorithm may

be customized to change the ranking of information types or to allow for weighting of the different attributes for prioritization, but not easily. This is merely a matter of adding more code.

The five dimension matrix weighs each attribute equally by putting the message in the matrix based upon the scale of each attribute. There is a weight given to each attribute based upon the order in which the search pattern progresses. In this case, the precedence type of Flash and the information types were considered the most important. Also, because of the data structures of this algorithm, the computer code used prevented the algorithm from being coded or tested.

D. CHAPTER SUMMARY

This chapter has described in detail the communications node model and the prioritization algorithms used on the messages in the communications node. The next chapter will examine the results of the model runs.

V. DATA ANALYSIS

The communications node model recorded several different data points, which included total number of messages sent, average wait time of messages before being sent and the average number of messages left in the queue at the end of the model run. The measure of performance used to judge the effectiveness of the algorigthms was the average wait time of a message. To ensure the data was normal, the model was run at nine different settings with fifty runs at each setting. These settings were chosen to look at the effect of the algorithms across a range of different situations. The settings are shown below in Table 1.

Table 1. Model Run Settings.

Run Num	**Msg Gen Rate 1**	**Msg Gen Rate 2**	**Ave Msg Size**	**Total**	**Comm Box Data Rate**
1	5	5	1000	10000	12500
2	5	5	1000	10000	11111
3	5	5	1000	10000	10526
4	50	5	1000	55000	88750
5	50	5	1000	55000	61111
6	50	5	1000	55000	57895
7	25	5	1000	30000	37500
8	25	5	1000	30000	33333
9	25	5	1000	30000	31579

Two-sample T tests and Analysis of Variance tests were run on the data. The null hypothesis was that the means of the two samples being compared

were equal. The alternative hypothesis was that the two sample means were not equal. This chapter discusses the analysis of the data given these suppositions. The data results are contained in Appendix C.

A. AVERAGE MESSAGE WAIT TIME COMPARISON

The average time a message waited before being transmitted was compared between the two algorithms at each of the nine run settings. At every run setting, the null hypothesis was rejected. The average wait time of a message in algorithm one was different than the average wait time for a message in algorithm two. In fact, at every run setting, the first algorithm (the modified FIFO queue) had a longer average wait time before the message was transmitted. This difference could not be attributed to the different run settings.

This would suggest that the prioritization/reprioritization algorithms do have an effect on the average wait time of a message. In addition, algorithm two, which reprioritized messages while they were in the queue, did have a positive effect on the communications node and reduced overall message wait times. A comparison of the means of the average wait times showed that on average, messages treated under algorithm one waited about one and one half times as long as those treated by algorithm two.

B. MESSAGE PRECEDENCE COMPARISONS

The data was compared between the two algorithms within each of the four message precedence categories: Flash, Immediate, Priority, and Routine. In the case of the Flash messages, the null hypothesis was rejected and the means of the two samples were considered to be different. In every case, the average wait time for Flash messages was less for algorithm one than algorithm two. The increase in wait time for algorithm two ranged from two

to eight times as much as the average wait under algorithm one: The average increase was about four times the algorithm one average wait. Some increase in average wait time for Flash messages was expected. Even though algorithm two takes into account other factors besides precedence, an increase to the magnitude of multiples of eight were not expected.

For Immediate precedence messages, the null hypothesis was rejected. The means of the two samples are different with algorithm one having lower average wait times than algorithm two in every case. The increase in average wait time ranged from 1.6 times the original average wait time to four times the average wait time. The average increase in wait times was approximately two times the original.

The results in the Priority precedence messages were more varied. In all cases, the null hypothesis was rejected and the sample means were found to be different. However, in three of the nine run settings, algorithm one was found to have longer average wait times while algorithm two had longer average wait times in the remaining six settings.

For the comparison of the Routine precedence messages, the null hypothesis was rejected in all cases. In every case, algorithm two had shorter average wait times than algorithm one. The increase in average wait times for algorithm one ranged from two to seven times the average wait times using algorithm two. The average increase was about five times the average wait time from algorithm two.

It is interesting to note that in every case, the prioritization/ reprioritization algorithms made a difference in the average wait time of messages before being transmitted. Indeed, as could have been predicted, the average wait time of the Flash messages was increased while the average wait time of the Routine messages was decreased with use of algorithm two. This would indicate that a prioritization/reprioritization algorithm can be

devised that would work in ADNS. One problem, however, is the four to eight times longer wait times of the Flash messages. The author believes a algorithm could be encoded that restricted the wait time of a flash message (immediate and priority as well) to a maximum time under normal to heavy loading while still improving the average wait time for routine and at least some priority messages.

C. PULL/PULL AND PUSH/PUSH COMPARISONS

The average wait for messages classified as user pull messages was compared between the two algorithms. As would be expected, the null hypothesis (the mean of sample one equals the mean of sample two) was rejected. In each run setting, user pull messages treated by algorithm one had a longer average wait time than those treated by algorithm two. This result was expected since only algorithm two actually factored into the prioritization if a message was in fact a user pull message. As explained earlier, the user pull messages were considered more of a priority than the producer push messages. A study of the means shows that the average wait time for the user pull messages under algorithm one was typically twice as long as the user pull messages under algorithm two.

The average wait time of producer push messages was also compared between algorithm one and algorithm two treatments. As was expected, the null hypothesis was rejected. Unexpectedly, the average wait time for algorithm one was longer than that for algorithm two. On average, the average wait for producer push messages under algorithm one was about one and one half times that for algorithm two. This is unusual in that algorithm two identified messages that were designated as producer push and lowered their priority vis-a-vis the user pull messages. Thus the expected action was that the producer push messages would have a longer average wait time

under algorithm two. It seems logical that the reduction in average wait time is due to the decrease in overall average message wait time between algorithm one and two. This has not been explored and provides an area for further research.

D. PULL VERSUS PUSH COMPARISONS

A comparison of the average wait times for user pull versus producer push messages was done to help ensure the algorithms operated correctly. In comparing the average wait time for user pull messages versus producer push messages under the algorithm one treatment, the null hypothesis was accepted. The average wait for user pull and producer push messages was statistically the same. This was expected since algorithm one did not differentiate between user pull and producer push messages.

For algorithm two, the null hypothesis was rejected. There was a difference between the average wait of the user pull messages and the producer push messages. In every run setting, the user pull message average wait was less than the producer push. Interestingly, the difference on the most lightly loaded runs was the smallest, with producer push message average wait about 1.1 times the wait for user pull messages. As the model loading increased, so did the difference in average wait times with the most heavily loaded runs having the average wait time for producer push messages 1.5 times the user pull message wait time. While the difference in average wait times was predicted based upon the successful run of algorithm two, the difference due to communications node loading was not forseen. The effect due to communications node loading is logical given the increased chance of a message having to wait as the loading increases and the priority of user pull messages over producer push.

E. CHAPTER SUMMARY

This chapter has examined the data results recorded after multiple model/algorithm runs. The data was moved into Microsoft Excel spreadsheets to ease data manipulation prior to the actual statistical tests being run in Minitab. On the whole, the results were as expected and allow for more study on what type of prioritization/reprioritization algorithms should be used.

VI. CONCLUSION

A. RESULTS

The results of the data analysis clearly show that the prioritization/ reprioritization algorithms do effect the average wait time of a message before it is transmitted. The effect is dependent on the algorithm used and what precepts it has written into the code. Overall, lower average wait time, the desired effect, was achieved and may indicate a route to more efficient use of the U.S. Navy's scarce communications resources.

B. LESSONS LEARNED

Several lessons learned were generated in the process of doing this thesis. First, more time would be allocated for generation of the computer code and most importantly, the troubleshooting of the code. Second, while not necessarily envisioned to be used during at the beginning of the thesis, additional data output would be written into the code at the beginning in order to minimize data manipulation at the end should the initial results require it. Third, use of computer code that more easily supports model generation and modification would allow the writer to focus more on the results of the model runs and reduce time spent on creating the model. This would also allow the coding of the third algorithm with an n-dimensional matrix holding the messages in priority order. Some suggestions might be OpNet, CommNet and SES/Workbench modeling codes.

C. RECOMMENDATIONS FOR FURTHER RESEARCH

Several areas from this thesis can be studied further to allow the U.S. Navy to take maximum advantage of these findings. First, more research

could be done on how much increase in the average wait time for Flash messages is allowable and how much does the average wait time for Routine messages (and Priority) change for each unit change in Flash message average wait time.

Second, the third algorithm mentioned above, or others similar to it, could be coded and tested to see what data structure/prioritization structure works most efficiently. As part of this research, more research on what factors should be used for the prioritization of a message as well as the relative weights of each factor should be conducted.

Third, research on how to encode the algorithms for ADNS could be done to ease utilization in the fleet. Other services should use this type of prioritization if the U.S. Navy does move towards this in order to avoid "stovepiping". Overall, software improvements to communications systems without the need to purchase additional equipment allows the military to continue to expand communication capabilities for less dollars.

APPENDIX A: COMMUNICATION NODE MODEL WITH ALGORITHM ONE

```
{Chris Halton
 Thesis Algorithm 4
 the Base Case (FIFO)
 last mod: 13 Aug 94}

unit THESIS1;

interface
  procedure PRIORITIZE3(L : integer; var outfile : text);
implementation

procedure PRIORITIZE3(L : integer; var outfile : text);
type FilePOINT = ^FileTYPE;
    FileTYPE = record
            FileInfoType : integer;
            FileUserPull : boolean;
            FileSize : integer;
            FileTimeEnter : real;
            Next : FilePOINT
          end;
    QueueRECORD = record
             Size : integer;
             Next : FilePOINT
           end;
    QueueTYPE = array[1..4] of QueueRECORD;
```

```
    AveWaitTYPE = array[1..4,1..14] of real;
    WaitTYPE = array[1..4,1..14] of real;
    SentTYPE = array[1..4,1..14] of integer;
    CountTYPE = array[1..4,1..14] of integer;
  var Queue : QueueTYPE;
    AveWaitMatrix : AveWaitTYPE;
    WaitMatrix : WaitTYPE;
    SentMatrix : SentTYPE;
    CountMatrix : CountTYPE;
    TotalMsgs, PushNumb, PushSent, PullNumb, PullSent,
TotalMsgSent,
    QueueLength : longint;
    InfoPriority, FSize, InfoType, InfoTemp, NumRate, NumRate2,
InfoType2,
    InfoType3, MsgSize, UsePull, Size1,
    i, j, InfoPri : integer;
    QueuePullUp, Msg1Up, Msg2Up, PullTest, MsgUp : boolean;
    NextTimeStep, ModelTime, AveWaitTime, PullWaitTime,
PullAveWait,
    MsgWaitTime, Msg1Time, Msg2Time, TotalWaitTime,
PushWaitTime,
    QueuePullTime, Msg1RN, Msg2RN, MsgTime, PushAveWait,
PercentSent,
    PercentQueue : real;
    TempPtr, Ptr : FilePOINT;


  const DataRate = 57895;
```

```
    ArrivalRate1 = 50;
    ArrivalRate2 = 5;
procedure MsgGen1(ModelTime : real; var MsgOneTime : real);
  begin
    Msg1RN := SYSTEM.Random;
    MsgOneTime := ModelTime - ((1/ArrivalRate1)*ln(Msg1RN));
  end;
procedure MsgGen2(ModelTime : real; var MsgTwoTime : real);
  begin
    Msg2RN := SYSTEM.Random;
    MsgTwoTime := ModelTime - ((1/ArrivalRate2)*ln(Msg2RN));
  end;
begin
  for i := 1 to 4 do begin
    Queue[i].Size := 0;
    for j := 1 to 14 do begin
      AveWaitMatrix[i,j] := 0.0;
      WaitMatrix[i,j] := 0.0;
      SentMatrix[i,j] := 0;
      CountMatrix[i,j] := 0;
    end;  {for}
  end;  {for}
  Randomize;
  ModelTime := 0.0;
  NextTimeStep := 0.0;
  QueueLength := 0;
  QueuePullTime := 0.0;
  TotalMsgSent := 0;
```

```
TotalWaitTime := 0.0;
AveWaitTime := 0.0;
UsePull := 0;
PullNumb := 0;
PullSent := 0;
PushNumb := 0;
PushSent := 0;
PullWaitTime := 0.0;
PullAveWait := 0.0;
PushWaitTime := 0.0;
PushAveWait := 0.0;
InfoPriority := 0;
FSize := 0;
Size1 := 0;
InfoType := 0;
InfoTemp := 0;
NumRate := 0;
NumRate2 := 0;
InfoType2 := 0;
MsgSize := 0;
InfoPri := 0;
MsgWaitTime := 0.0;
MsgTime := 0.0;
MsgGen1(ModelTime,Msg1Time);
MsgGen2(ModelTime,Msg2Time);
while (ModelTime < 1000.0) and (QueueLength < 10000) do begin
  {writeln('Start '); }
  MsgUp := False;
```

```
Msg1Up := False;
Msg2Up := False;
QueuePullUp := False;
if (Msg1Time >= ModelTime) and (Msg2Time >= ModelTime) and
  (Msg1Time < Msg2Time) then begin
 NextTimeStep := Msg1Time;
end  {if}
else if(Msg1Time >= ModelTime) and (Msg2Time >= ModelTime) and
  (Msg2Time < Msg1Time) then  begin
 NextTimeStep := Msg2Time
end  { else}
else if (Msg1Time >= ModelTime) and (Msg2Time < ModelTime) then
begin
 NextTimeStep := Msg1Time;
end  {else if}
else if (Msg1Time < ModelTime) and (Msg2Time >= ModelTime) then
begin
 NextTimeStep := Msg2Time
end;  {else if}
if (QueuePullTime <> 0.0) then begin
 if (QueuePullTime > ModelTime) and (QueuePullTime <
NextTimeStep)
 then begin
  NextTimeStep := QueuePullTime
 end; {if}
end; {if}

if (NextTimeStep = Msg1Time) then begin
```

```
 Msg1Up := True
end; {if}
if (NextTimeStep = Msg2Time) then begin
 Msg2Up := True
end; {if}
if (NextTimeStep = QueuePullTime) then begin
 QueuePullUp := True
end; {if}
ModelTime := NextTimeStep;
if Msg1Up then begin
 {writeln('Msg 1 ');}
 InfoPriority := Random(4);
 InfoPri := InfoPriority + 1;
 Size1 := Random(2000);
 FSize := Size1 + 1; {to avoid zero}
 InfoType := Random(14);
 InfoTemp := InfoType + 1; {to get into queue}
 UsePull := Random(4);
 if (UsePull = 0) then begin
  PullNumb := PullNumb + 1;
  PullTest := True;
 end
 else begin
  PushNumb := PushNumb + 1;
  PullTest := False;
 end;
 CountMatrix[InfoPri,InfoTemp] := CountMatrix[InfoPri,InfoTemp] +
1;
```

```
        if (QueueLength = 0) and (QueuePullTime <= ModelTime) then
begin
         QueuePullUp := True;
         QueuePullTime := ModelTime;
         MsgUp := True;
        end   {if}
        else begin    {place in queue}
         NumRate := InfoPriority + 1;   {to adjust to queue numbers}
         if Queue[NumRate].Size = 0 then begin
          new(Queue[NumRate].Next);
          Queue[NumRate].Next^.FileInfoType := InfoType;
          Queue[NumRate].Next^.FileSize := FSize;
          Queue[NumRate].Next^.FileTimeEnter := ModelTime;
          if PullTest then begin
           Queue[NumRate].Next^.FileUserPull := True;
          end
          else begin
           Queue[NumRate].Next^.FileUserPull := False;
          end;
          Queue[NumRate].Next^.Next := nil;
          Queue[NumRate].Size := 1;
         end  {if}
         else begin
          Ptr := Queue[NumRate].Next;
          while Ptr^.Next <> nil do begin
           Ptr := Ptr^.Next;
          end;  {while}
          new(Ptr^.Next);
```

```
    Ptr := Ptr^.Next;
    Ptr^.FileInfoType := InfoType;
    Ptr^.FileSize := FSize;
    Ptr^.FileTimeEnter := ModelTime;
    Ptr^.FileUserPull := PullTest;
    Ptr^.Next := nil;
    Queue[NumRate].Size := Queue[NumRate].Size + 1;
   end;  {else}
   QueueLength := QueueLength + 1;
  end;  {else}
  Msg1Up := False;
  MsgGen1(ModelTime,Msg1Time);
 end;  {if}
 if Msg2Up then begin
  {writeln('          Msg2');}
  InfoPriority := Random(4);
  InfoPri := InfoPriority + 1; {to adjust to queue numbers}
  Size1 := Random(2000);
  FSize := Size1 + 1;  {to avoid zeros}
  InfoType := Random(14);
  InfoTemp := InfoType + 1;  {to adjust to queue numbers}
  UsePull := Random(4);
  if (UsePull = 3) then begin
   PushNumb := PushNumb + 1;
   PullTest := False;
  end
  else begin
   PullNumb := PullNumb + 1;
```

```
      PullTest := True;
    end;
    CountMatrix[InfoPri,InfoTemp] := CountMatrix[InfoPri,InfoTemp] +
1;
    if (QueueLength = 0) and (QueuePullTime <= ModelTime) and
     (not QueuePullUp) then begin
      QueuePullUp := True;
      QueuePullTime := ModelTime;
      MsgUp := True;
    end  {if}
    else begin    {place in queue}
      NumRate := InfoPriority + 1;   {to adjust to queue numbers}
      if Queue[NumRate].Size = 0 then begin
        new(Queue[NumRate].Next);
        Queue[NumRate].Next^.FileInfoType := InfoType;
        Queue[NumRate].Next^.FileSize := FSize;
        Queue[NumRate].Next^.FileTimeEnter := ModelTime;
        Queue[NumRate].Next^.FileUserPull := PullTest;
        Queue[NumRate].Next^.Next := nil;
        Queue[NumRate].Size := 1;
      end  {if}
      else begin
        Ptr := Queue[NumRate].Next;
        while Ptr^.Next <> nil do begin
          Ptr := Ptr^.Next;
        end;  {while}
        new(Ptr^.Next);
        Ptr := Ptr^.Next;
```

```
    Ptr^.FileInfoType := InfoType;
    Ptr^.FileSize := FSize;
    Ptr^.FileTimeEnter := ModelTime;
    Ptr^.FileUserPull := PullTest;
    Ptr^.Next := nil;
    Queue[NumRate].Size := Queue[NumRate].Size + 1;
   end;  {else}
   QueueLength := QueueLength + 1;
  end; {else}
  Msg2Up := False;
  MsgGen2(ModelTime,Msg2Time);
 end;  {if}
 if QueuePullUp then begin
  {writeln('      QueuePullUp');}
  if (QueueLength = 0) and (MsgUp = False) then begin
   QueuePullUp := False;
   {writeln('  ModelTime =  ',ModelTime,'           QueuePull False');}
  end  {if}
  else if (QueueLength = 0) and (MsgUp = True) then begin
   {writeln('       QueueLength = 0  ');}
   MsgTime := FSize/DataRate;
   QueuePullTime := ModelTime + MsgTime;
   if PullTest then begin
    PullSent := PullSent + 1;
    PullAveWait := PullWaitTime/PullSent;
   end
   else begin
    PushSent := PushSent + 1;
```

```
          PushAveWait := PushWaitTime/PushSent;
        end;
        NumRate := InfoPriority + 1;  {to get queue numbers}
        NumRate2 := InfoType + 1;  {to get queue numbers}
        SentMatrix[NumRate,NumRate2] :=
SentMatrix[NumRate,NumRate2] + 1;
        AveWaitMatrix[NumRate,NumRate2] :=
WaitMatrix[NumRate,NumRate2]/
          SentMatrix[NumRate,NumRate2];
        CountMatrix[NumRate,NumRate2] :=
CountMatrix[NumRate,NumRate2] - 1;
        QueuePullUp := False;
      end  { if}
      else if (QueueLength > 0) then begin
        {writeln('QueueLength  =  ',QueueLength);}
        if Queue[1].Size > 0 then begin
          MsgSize := Queue[1].Next^.FileSize;
          MsgTime := (MsgSize/DataRate);
          MsgWaitTime := ModelTime - Queue[1].Next^.FileTimeEnter;
          InfoType2 := Queue[1].Next^.FileInfoType;
          InfoType3 := InfoType2 + 1;  {to get queue numbers}
          if Queue[1].Next^.FileUserPull then begin
            PullWaitTime := PullWaitTime + MsgWaitTime;
            PullSent := PullSent + 1;
            PullAveWait := PullWaitTime/PullSent;
          end
          else begin
            PushWaitTime := PushWaitTime + MsgWaitTime;
```

```
    PushSent := PushSent + 1;
    PushAveWait := PushWaitTime/PushSent;
   end;
   SentMatrix[1,InfoType3] := SentMatrix[1,InfoType3] + 1;
   CountMatrix[1,InfoType3] := CountMatrix[1,InfoType3] - 1;
   WaitMatrix[1,InfoType3] := WaitMatrix[1,InfoType3] +
MsgWaitTime;
   AveWaitMatrix[1,InfoType3] :=
WaitMatrix[1,InfoType3]/SentMatrix[1,InfoType3];
   QueuePullTime := ModelTime + MsgTime;
   Queue[1].Size := Queue[1].Size - 1;
   QueueLength := QueueLength - 1;
   if Queue[1].Next^.Next <> nil then begin
    TempPtr := Queue[1].Next;
    Queue[1].Next := Queue[1].Next^.Next;
    dispose(TempPtr);
    TempPtr := nil;
   end  {if}
   else if Queue[1].Next^.Next = nil then begin
    dispose(Queue[1].Next);
    Queue[1].Next := nil;
   end;
  end  {if}
  else if Queue[2].Size > 0 then begin
   MsgSize := Queue[2].Next^.FileSize;
   MsgTime := (MsgSize/DataRate);
   MsgWaitTime := ModelTime - Queue[2].Next^.FileTimeEnter;
   InfoType2 := Queue[2].Next^.FileInfoType;
```

```
InfoType3 := InfoType2 + 1;  {to get queue numbers}
if Queue[2].Next^.FileUserPull then begin
  PullWaitTime := PullWaitTime + MsgWaitTime;
  PullSent := PullSent + 1;
  PullAveWait := PullWaitTime/PullSent;
end
else begin
  PushWaitTime := PushWaitTime + MsgWaitTime;
  PushSent := PushSent + 1;
  PushAveWait := PushWaitTime/PushSent;
end;
SentMatrix[2,InfoType3] := SentMatrix[2,InfoType3] + 1;
CountMatrix[2,InfoType3] := CountMatrix[2,InfoType3] - 1;
WaitMatrix[2,InfoType3] := WaitMatrix[2,InfoType3] +
MsgWaitTime;
AveWaitMatrix[2,InfoType3] :=
WaitMatrix[2,InfoType3]/SentMatrix[2,InfoType3];
QueuePullTime := ModelTime + MsgTime;
Queue[2].Size := Queue[2].Size - 1;
QueueLength := QueueLength - 1;
if Queue[2].Next^.Next <> nil then begin
  TempPtr := Queue[2].Next;
  Queue[2].Next := Queue[2].Next^.Next;
  dispose(TempPtr);
  TempPtr := nil;
end  {if}
else if Queue[2].Next^.Next = nil then begin
  dispose(Queue[2].Next);
```

```
    Queue[2].Next := nil;
   end;
  end {if}
  else if Queue[3].Size > 0 then begin
   MsgSize := Queue[3].Next^.FileSize;
   MsgTime := (MsgSize/DataRate);
   MsgWaitTime := ModelTime - Queue[3].Next^.FileTimeEnter;
   InfoType2 := Queue[3].Next^.FileInfoType;
   InfoType3 := InfoType2 + 1;  {to get queue numbers}
   if Queue[3].Next^.FileUserPull then begin
    PullWaitTime := PullWaitTime + MsgWaitTime;
    PullSent := PullSent + 1;
    PullAveWait := PullWaitTime/PullSent;
   end
   else begin
    PushWaitTime := PushWaitTime + MsgWaitTime;
    PushSent := PushSent + 1;
    PushAveWait := PushWaitTime/PushSent;
   end;
   SentMatrix[3,InfoType3] := SentMatrix[3,InfoType3] + 1;
   CountMatrix[3,InfoType3] := CountMatrix[3,InfoType3] - 1;
   WaitMatrix[3,InfoType3] := WaitMatrix[3,InfoType3] +
MsgWaitTime;
   AveWaitMatrix[3,InfoType3] :=
WaitMatrix[3,InfoType3]/SentMatrix[3,InfoType3];
   QueuePullTime := ModelTime + MsgTime;
   Queue[3].Size := Queue[3].Size - 1;
   QueueLength := QueueLength - 1;
```

```
    if Queue[3].Next^.Next <> nil then begin
      TempPtr := Queue[3].Next;
      Queue[3].Next := Queue[3].Next^.Next;
      dispose(TempPtr);
      TempPtr := nil;
    end  {if}
    else if Queue[3].Next^.Next = nil then begin
      dispose(Queue[3].Next);
      Queue[3].Next := nil;
    end;
  end  {if}
  else if Queue[4].Size > 0 then begin
    MsgSize := Queue[4].Next^.FileSize;
    MsgTime := (MsgSize/DataRate);
    MsgWaitTime := ModelTime - Queue[4].Next^.FileTimeEnter;
    InfoType2 := Queue[4].Next^.FileInfoType;
    InfoType3 := InfoType2 + 1;  {to get queue numbers}
    if Queue[4].Next^.FileUserPull then begin
      PullWaitTime := PullWaitTime + MsgWaitTime;
      PullSent := PullSent + 1;
      PullAveWait := PullWaitTime/PullSent;
    end
    else begin
      PushWaitTime := PushWaitTime + MsgWaitTime;
      PushSent := PushSent + 1;
      PushAveWait := PushWaitTime/PushSent;
    end;
    SentMatrix[4,InfoType3] := SentMatrix[4,InfoType3] + 1;
```

```
            CountMatrix[4,InfoType3] := CountMatrix[4,InfoType3] - 1;
            WaitMatrix[4,InfoType3] := WaitMatrix[4,InfoType3] +
MsgWaitTime;
            AveWaitMatrix[4,InfoType3] :=
WaitMatrix[4,InfoType3]/SentMatrix[4,InfoType3];
            QueuePullTime := ModelTime + MsgTime;
            Queue[4].Size := Queue[4].Size - 1;
            QueueLength := QueueLength - 1;
            if Queue[4].Next^.Next <> nil then begin
             TempPtr := Queue[4].Next;
             Queue[4].Next := Queue[4].Next^.Next;
             dispose(TempPtr);
             TempPtr := nil;
            end  {if}
            else if Queue[4].Next^.Next = nil then begin
             dispose(Queue[4].Next);
             Queue[4].Next := nil;
            end;
            QueuePullUp := False;
           end;  {if}
          end;  {else}
         end;  {if}
        end;  {while}
        for i := 1 to 4 do begin
         while Queue[i].Next <> nil do begin
          Ptr := Queue[i].Next;
          Queue[i].Next := Queue[i].Next^.Next;
          dispose(Ptr);
```

```
    Ptr := nil;
  end;  {while}
end;  {for}
for i := 1 to 4 do begin
  for j := 1 to 14 do begin
    TotalMsgSent := TotalMsgSent + SentMatrix[i,j];
    TotalWaitTime := TotalWaitTime + WaitMatrix[i,j];
  end; {for}
end;  {for}
{writeln('QueueLength  = ',QueueLength);}
TotalMsgs := QueueLength + TotalMsgSent;
PercentSent := TotalMsgSent/TotalMsgs;
PercentQueue := QueueLength/TotalMsgs;
AveWaitTime := TotalWaitTime/TotalMsgSent;
write(outfile,'1,');
write(outfile,L,',');
Write(outfile,DataRate,',');
Write(outfile,ArrivalRate1,',');
Write(outfile,ArrivalRate2,',');
write(outfile,ModelTime:5:3,',');
write(outfile,QueueLength,',');
write(outfile,TotalMsgSent,',');
write(outfile,AveWaitTime:5:3,',');
write(outfile,TotalMsgs,',');
write(outfile,PercentSent:5:3,',');
write(outfile,PercentQueue:5:3,',');
write(outfile,PullNumb,',');
write(outfile,PullSent,',');
```

```
write(outfile,PullAveWait:5:3,',');
write(outfile,PushNumb,',');
write(outfile,PushSent,',');
write(outfile,PushAveWait:5:3,',');
write(outfile,SentMatrix[1,1],',');
write(outfile,AveWaitMatrix[1,1]:5:3,',');
write(outfile,CountMatrix[1,1],',');
write(outfile,SentMatrix[1,2],',');
write(outfile,AveWaitMatrix[1,2]:5:3,',');
write(outfile,CountMatrix[1,2],',');
write(outfile,SentMatrix[1,3],',');
write(outfile,AveWaitMatrix[1,3]:5:3,',');
write(outfile,CountMatrix[1,3],',');
write(outfile,SentMatrix[1,4],',');
write(outfile,AveWaitMatrix[1,4]:5:3,',');
write(outfile,CountMatrix[1,4],',');
write(outfile,SentMatrix[1,5],',');
write(outfile,AveWaitMatrix[1,5]:5:3,',');
write(outfile,CountMatrix[1,5],',');
write(outfile,SentMatrix[1,6],',');
write(outfile,AveWaitMatrix[1,6]:5:3,',');
write(outfile,CountMatrix[1,6],',');
write(outfile,SentMatrix[1,7],',');
write(outfile,AveWaitMatrix[1,7]:5:3,',');
write(outfile,CountMatrix[1,7],',');
write(outfile,SentMatrix[1,8],',');
write(outfile,AveWaitMatrix[1,8]:5:3,',');
write(outfile,CountMatrix[1,8],',');
```

```
write(outfile,SentMatrix[1,9],',');
write(outfile,AveWaitMatrix[1,9]:5:3,',');
write(outfile,CountMatrix[1,9],',');
write(outfile,SentMatrix[1,10],',');
write(outfile,AveWaitMatrix[1,10]:5:3,',');
write(outfile,CountMatrix[1,10],',');
write(outfile,SentMatrix[1,11],',');
write(outfile,AveWaitMatrix[1,11]:5:3,',');
write(outfile,CountMatrix[1,11],',');
write(outfile,SentMatrix[1,12],',');
write(outfile,AveWaitMatrix[1,12]:5:3,',');
write(outfile,CountMatrix[1,12],',');
write(outfile,SentMatrix[1,13],',');
write(outfile,AveWaitMatrix[1,13]:5:3,',');
write(outfile,CountMatrix[1,13],',');
write(outfile,SentMatrix[1,14],',');
write(outfile,AveWaitMatrix[1,14]:5:3,',');
write(outfile,CountMatrix[1,14],',');
write(outfile,SentMatrix[2,1],',');
write(outfile,AveWaitMatrix[2,1]:5:3,',');
write(outfile,CountMatrix[2,1],',');
write(outfile,SentMatrix[2,2],',');
write(outfile,AveWaitMatrix[2,2]:5:3,',');
write(outfile,CountMatrix[2,2],',');
write(outfile,SentMatrix[2,3],',');
write(outfile,AveWaitMatrix[2,3]:5:3,',');
write(outfile,CountMatrix[2,3],',');
write(outfile,SentMatrix[2,4],',');
```

```
write(outfile,AveWaitMatrix[2,4]:5:3,',');
write(outfile,CountMatrix[2,4],',');
write(outfile,SentMatrix[2,5],',');
write(outfile,AveWaitMatrix[2,5]:5:3,',');
write(outfile,CountMatrix[2,5],',');
write(outfile,SentMatrix[2,6],',');
write(outfile,AveWaitMatrix[2,6]:5:3,',');
write(outfile,CountMatrix[2,6],',');
write(outfile,SentMatrix[2,7],',');
write(outfile,AveWaitMatrix[2,7]:5:3,',');
write(outfile,CountMatrix[2,7],',');
write(outfile,SentMatrix[2,8],',');
write(outfile,AveWaitMatrix[2,8]:5:3,',');
write(outfile,CountMatrix[2,8],',');
write(outfile,SentMatrix[2,9],',');
write(outfile,AveWaitMatrix[2,9]:5:3,',');
write(outfile,CountMatrix[2,9],',');
write(outfile,SentMatrix[2,10],',');
write(outfile,AveWaitMatrix[2,10]:5:3,',');
write(outfile,CountMatrix[2,10],',');
write(outfile,SentMatrix[2,11],',');
write(outfile,AveWaitMatrix[2,11]:5:3,',');
write(outfile,CountMatrix[2,11],',');
write(outfile,SentMatrix[2,12],',');
write(outfile,AveWaitMatrix[2,12]:5:3,',');
write(outfile,CountMatrix[2,12],',');
write(outfile,SentMatrix[2,13],',');
write(outfile,AveWaitMatrix[2,13]:5:3,',');
```

```
write(outfile,CountMatrix[2,13],',');
write(outfile,SentMatrix[2,14],',');
write(outfile,AveWaitMatrix[2,14]:5:3,',');
write(outfile,CountMatrix[2,14],',');
write(outfile,SentMatrix[3,1],',');
write(outfile,AveWaitMatrix[3,1]:5:3,',');
write(outfile,CountMatrix[3,1],',');
write(outfile,SentMatrix[3,2],',');
write(outfile,AveWaitMatrix[3,2]:5:3,',');
write(outfile,CountMatrix[3,2],',');
write(outfile,SentMatrix[3,3],',');
write(outfile,AveWaitMatrix[3,3]:5:3,',');
write(outfile,CountMatrix[3,3],',');
write(outfile,SentMatrix[3,4],',');
write(outfile,AveWaitMatrix[3,4]:5:3,',');
write(outfile,CountMatrix[3,4],',');
write(outfile,SentMatrix[3,5],',');
write(outfile,AveWaitMatrix[3,5]:5:3,',');
write(outfile,CountMatrix[3,5],',');
write(outfile,SentMatrix[3,6],',');
write(outfile,AveWaitMatrix[3,6]:5:3,',');
write(outfile,CountMatrix[3,6],',');
write(outfile,SentMatrix[3,7],',');
write(outfile,AveWaitMatrix[3,7]:5:3,',');
write(outfile,CountMatrix[3,7],',');
write(outfile,SentMatrix[3,8],',');
write(outfile,AveWaitMatrix[3,8]:5:3,',');
write(outfile,CountMatrix[3,8],',');
```

```
write(outfile,SentMatrix[3,9],',');
write(outfile,AveWaitMatrix[3,9]:5:3,',');
write(outfile,CountMatrix[3,9],',');
write(outfile,SentMatrix[3,10],',');
write(outfile,AveWaitMatrix[3,10]:5:3,',');
write(outfile,CountMatrix[3,10],',');
write(outfile,SentMatrix[3,11],',');
write(outfile,AveWaitMatrix[3,11]:5:3,',');
write(outfile,CountMatrix[3,11],',');
write(outfile,SentMatrix[3,12],',');
write(outfile,AveWaitMatrix[3,12]:5:3,',');
write(outfile,CountMatrix[3,12],',');
write(outfile,SentMatrix[3,13],',');
write(outfile,AveWaitMatrix[3,13]:5:3,',');
write(outfile,CountMatrix[3,13],',');
write(outfile,SentMatrix[3,14],',');
write(outfile,AveWaitMatrix[3,14]:5:3,',');
write(outfile,CountMatrix[3,14],',');
write(outfile,SentMatrix[4,1],',');
write(outfile,AveWaitMatrix[4,1]:5:3,',');
write(outfile,CountMatrix[4,1],',');
write(outfile,SentMatrix[4,2],',');
write(outfile,AveWaitMatrix[4,2]:5:3,',');
write(outfile,CountMatrix[4,2],',');
write(outfile,SentMatrix[4,3],',');
write(outfile,AveWaitMatrix[4,3]:5:3,',');
write(outfile,CountMatrix[4,3],',');
write(outfile,SentMatrix[4,4],',');
```

```
write(outfile,AveWaitMatrix[4,4]:5:3,',');
write(outfile,CountMatrix[4,4],',');
write(outfile,SentMatrix[4,5],',');
write(outfile,AveWaitMatrix[4,5]:5:3,',');
write(outfile,CountMatrix[4,5],',');
write(outfile,SentMatrix[4,6],',');
write(outfile,AveWaitMatrix[4,6]:5:3,',');
write(outfile,CountMatrix[4,6],',');
write(outfile,SentMatrix[4,7],',');
write(outfile,AveWaitMatrix[4,7]:5:3,',');
write(outfile,CountMatrix[4,7],',');
write(outfile,SentMatrix[4,8],',');
write(outfile,AveWaitMatrix[4,8]:5:3,',');
write(outfile,CountMatrix[4,8],',');
write(outfile,SentMatrix[4,9],',');
write(outfile,AveWaitMatrix[4,9]:5:3,',');
write(outfile,CountMatrix[4,9],',');
write(outfile,SentMatrix[4,10],',');
write(outfile,AveWaitMatrix[4,10]:5:3,',');
write(outfile,CountMatrix[4,10],',');
write(outfile,SentMatrix[4,11],',');
write(outfile,AveWaitMatrix[4,11]:5:3,',');
write(outfile,CountMatrix[4,11],',');
write(outfile,SentMatrix[4,12],',');
write(outfile,AveWaitMatrix[4,12]:5:3,',');
write(outfile,CountMatrix[4,12],',');
write(outfile,SentMatrix[4,13],',');
write(outfile,AveWaitMatrix[4,13]:5:3,',');
```

```
  write(outfile,CountMatrix[4,13],',');
  write(outfile,SentMatrix[4,14],',');
  write(outfile,AveWaitMatrix[4,14]:5:3,',');
  writeln(outfile,CountMatrix[4,14],',');
end;  {Prioritize3}
begin
end.  {Thesis1}
```

APPENDIX B: COMMUNICATION NODE MODEL WITH ALGORITHM TWO

```
{Chris Halton
 Thesis2 Algorithm w/ Model and Matrix Counters
 Last updated: 13 Aug 94}

unit THESIS2C;
interface
 procedure PRIORITIZE(L : integer; var outfile : text);
implementation
procedure PRIORITIZE(L : integer; var outfile : text);
type FilePOINT = ^FileTYPE;
    FileTYPE = record
            FileInfoPriority : integer;
            FileInfoType : integer;
            FileUserPull : boolean;
            FileSize : integer;
            FileTimeEnter : real;
            Next : FilePOINT
           end;
    QueueRECORD = record
             Full : boolean;
             PrevNum : integer;
             NextNum : integer;
             Next : FilePOINT
            end;

    AveWaitTYPE = array[1..4,1..15] of real;
```

```
    WaitTYPE = array[1..4,1..15] of real;
    SentTYPE = array[1..4,1..15] of integer;
    CountTYPE = array[1..4,1..15] of integer;
    QueueTYPE = array[1..2200] of QueueRECORD;
  var Queue : QueueTYPE;
    MsgAveWaitMatrix : AveWaitTYPE;
    MsgWaitMatrix : WaitTYPE;
    MsgSentMatrix : SentTYPE;
    CountMatrix : CountTYPE;
    QueueLength, MsgSent, PushNumb, PushSent, PullNumb, PullSent,
    TotalMsg, Counter : longint;
    InfoPriority, Size, Size1, UserPull, UserPull2, Priority,
    NumRate, QueueBegin, QueueTemp, MsgSize, TempInfoPri, Temp,
InfoType,
    TempSize, QueuePtr, i, j, K, Temp2, QueueEnd, InfoTemp,
    PriTemp, InfoTemp2, PriTemp2, InfoTemp3, PriTemp3, InfoTemp4,
PriTemp4,
    InfoTemp1, PriTemp1, TempInfoType, Size2 : integer;
    QueueReDoUp, QueuePullUp, MsgUp, Msg1Up, Msg2Up, MsgUp1,
MsgUp2,
    PullTest, TempUsePull : boolean;
    NextTimeStep, TempTime, HoldTime, ModelTime, AveWait,
AveQueueTime,
    QueueWaitTime, MsgWaitTime, Msg1Time, Msg2Time,
QueueReDoTime,
    QueuePullTime, Msg1RN, Msg2RN, MsgTime, PullWaitTime,
PullAveWait,
```

```
        PushWaitTime, PushAveWait, PercentSent, PercentQueue, TempAve
: real;
        TempPtr,TempPtr2, Ptr : FilePOINT;
    const DataRate = 57895;
        ArrivalRate1 = 50;
        ArrivalRate2 = 5;
    procedure MsgGen1(ModelTime : real; var MsgOneTime : real);
     begin
      Msg1RN := SYSTEM.Random;
      MsgOneTime := ModelTime - ((1/ArrivalRate1)*ln(Msg1RN));
     end;
    procedure MsgGen2(ModelTime : real; var MsgTwoTime : real);
     begin
      Msg2RN := SYSTEM.Random;
      MsgTwoTime := ModelTime - ((1/ArrivalRate2)*ln(Msg2RN));
     end;
    procedure QueueReDoTimeGen(ModelTime : real; var QueueReDoTime :
real);
     begin
      QueueReDoTime := ModelTime + 5.0;
     end;
    procedure NumericalOrder(NRate : integer; var QBegin, QEnd : integer;
                  var Queue2 : QueueTYPE);
     var QTemp : integer;
     begin
      if QBegin = QEnd then begin
       if NRate < QBegin then begin
        Queue2[NRate].NextNum := QBegin;
```

```
    Queue2[QBegin].PrevNum := NRate;
    QBegin := NRate;
  end {if}
  else if QEnd < NRate then begin
    Queue2[QEnd].NextNum := NRate;
    Queue2[NRate].PrevNum := QEnd;
    QEnd := NRate;
  end; {else if}
end {if}
else if QBegin <> QEnd then begin
  if (QBegin = 0) and (QEnd = maxint) then begin
    QBegin := NRate;
    QEnd := NRate;
  end {if}
  else if NRate < QBegin then begin
    Queue2[NRate].NextNum := QBegin;
    Queue2[QBegin].PrevNum := NRate;
    QBegin := NRate;
  end {if}
  else if QEnd < NRate then begin
    Queue2[QEnd].NextNum := NRate;
    Queue2[NRate].PrevNum := QEnd;
    QEnd := NRate;
  end {else if}
  else if (QBegin < NRate) and (NRate < QEnd) then begin
    QTemp := QBegin;
    while Queue2[QTemp].NextNum < NRate do begin
      QTemp := Queue2[QTemp].NextNum;
```

```
      end;  {while}
      Queue2[NRate].NextNum := Queue2[QTemp].NextNum;
      Queue2[NRate].PrevNum := QTemp;
      Queue2[QTemp].NextNum := NRate;
      Queue2[Queue2[NRate].NextNum].PrevNum := NRate;
     end;  {else if}
    end; {else if}
   end; {Numerical Order}
  procedure QueueFullFalse(var Queue1 : QueueTYPE; NumRat : integer;
                 var Q1Begin, Q1End : integer; IPriority,
                 IType : integer; UPull : boolean;
                 Size3 : integer; MTime : real);
   begin
    new(Queue1[NumRat].Next);
    Queue1[NumRat].Next^.FileInfoPriority := IPriority;
    Queue1[NumRat].Next^.FileInfoType := IType;
    Queue1[NumRat].Next^.FileUserPull := UPull;
    Queue1[NumRat].Next^.FileSize := Size3;
    Queue1[NumRat].Next^.FileTimeEnter := MTime;
    Queue1[NumRat].Next^.Next := nil;
    Queue1[NumRat].Full := True;
    NumericalOrder(NumRat,Q1Begin,Q1End,Queue1);
   end;  {QueueFullFalse}
  procedure QueueFullTrue(NumRate1 : integer; var Queue3 :
QueueTYPE;
                 IPri, InfoTy : integer; UserP : boolean;
                 Size4 : integer; ModTime : real);
```

```
var Ptr1 : FilePoint;
begin
  Ptr1 := Queue3[NumRate1].Next;
  while Ptr1^.Next <> nil do begin
    Ptr1 := Ptr1^.Next;
  end;  {while}
  new (Ptr1^.Next);
  Ptr1 := Ptr1^.Next;
  Ptr1^.FileInfoPriority := IPri;
  Ptr1^.FileInfoType := InfoTy;
  Ptr1^.FileUserPull := UserP;
  Ptr1^.FileSize := Size4;
  Ptr1^.FileTimeEnter := ModTime;
  Ptr1^.Next := nil;
end;  {else}
begin
 for i := 1 to 2200 do begin
  Queue[i].PrevNum := 0;
  Queue[i].NextNum :=0;
  Queue[i].Full := False;
  Queue[i].Next := nil;
 end;  {for}
 for i := 1 to 4 do begin
  for j:= 1 to 15 do begin
    MsgAveWaitMatrix[i,j] := 0.0;
    MsgWaitMatrix[i,j] := 0.0;
    MsgSentMatrix[i,j] := 0;
    CountMatrix[i,j] := 0;
```

```
  end;  {for}
end;  {for}
Randomize;
ModelTime := 0.0;
NextTimeStep := 0.0;
QueueLength := 0;
QueuePullTime := 0.0;
QueueBegin := 0;
QueueEnd := maxint;
Size := 0;
Size1 := 0;
Size2 := 0;
InfoPriority := 0;
InfoType := 0;
UserPull := 0;
UserPull2 := 0;
InfoTemp := 0;
PriTemp := 0;
InfoTemp1 := 0;
PriTemp2 := 0;
InfoTemp2 := 0;
PriTemp2 := 0;
InfoTemp3 := 0;
PriTemp3 := 0;
InfoTemp4 := 0;
PriTemp4 := 0;
TempAve := 0.0;
Priority := 0;
```

```
MsgSent := 0;
AveWait := 0.0;
TotalMsg := 0;
PullNumb := 0;
PullSent := 0;
PullWaitTime := 0.0;
PullAveWait := 0.0;
PushNumb := 0;
PushSent := 0;
PushWaitTime := 0.0;
PushAveWait := 0.0;
PercentSent := 0.0;
PercentQueue := 0.0;
QueueWaitTime := 0.0;
MsgGen1(ModelTime,Msg1Time);
MsgGen2(ModelTime,Msg2Time);
QueueReDoTimeGen(ModelTime,QueueReDoTime);
Counter := 0;
while (ModelTime < 1000.0) and (QueueLength < 10000) and (Counter
<200000) do begin
 {writeln('Start '); }
 MsgUp := False;
 Msg1Up := False;
 Msg2Up := False;
 MsgUp1 := False;
 MsgUp2 := False;
 QueueReDoUp := False;
 QueuePullUp := False;
```

```
        if (Msg1Time >= ModelTime) and (Msg2Time >= ModelTime) and
          (Msg1Time < Msg2Time) then begin
         NextTimeStep := Msg1Time;
        end  {if}
        else if(Msg1Time >= ModelTime) and (Msg2Time >= ModelTime) and
          (Msg2Time < Msg1Time) then  begin
         NextTimeStep := Msg2Time
        end  { else}
        else if (Msg1Time >= ModelTime) and (Msg2Time < ModelTime) then
begin
         NextTimeStep := Msg1Time;
        end  {else if}
        else if (Msg1Time < ModelTime) and (Msg2Time >= ModelTime) then
begin
         NextTimeStep := Msg2Time
        end;  {else if}
        if (QueueReDoTime >= ModelTime) and (QueueReDoTime <
NextTimeStep)
        then begin
         NextTimeStep := QueueReDoTime
        end;  {if}
        if (QueuePullTime <> 0.0) then begin
         if (QueuePullTime > ModelTime) and (QueuePullTime <
NextTimeStep)
         then begin
          NextTimeStep := QueuePullTime
         end;  {if}
```

```
end; {if}
if (NextTimeStep = Msg1Time) then begin
 Msg1Up := True
end; {if}
if (NextTimeStep = Msg2Time) then begin
 Msg2Up := True
end; {if}
if (NextTimeStep = QueueReDoTime) then begin
 QueueReDoUp := True
end; {if}
if (NextTimeStep = QueuePullTime) then begin
 QueuePullUp := True
end; {if}
ModelTime := NextTimeStep;
if Msg1Up then begin
 {writeln('   Msg1 Up ',ModelTime:5:3); }
 InfoPriority := Random(4);
 InfoType := Random(14);
 Size1 := Random(2000);
 Size := Size1 + 1;
 UserPull := Random(4);
 if (UserPull = 0) then begin
  PullNumb := PullNumb + 1;
  PullTest := True;
 end
 else begin
  PushNumb := PushNumb + 1;
  PullTest := False;
```

```
        end;
        InfoTemp2 := InfoType + 1;  {to get queue numbering}
        PriTemp2 := InfoPriority + 1;  { to get queue numbering}
        CountMatrix[PriTemp2,InfoTemp2] :=
CountMatrix[PriTemp2,InfoTemp2] + 1;
        if (QueueLength = 0) and (QueuePullTime <= ModelTime) then
begin
         QueuePullUp := True;
         QueuePullTime := ModelTime;
         MsgUp := True;
        end  {if}
        else begin   {place in queue}
         Priority := InfoPriority + Size + InfoType;
         if PullTest then begin
          if Priority > 30 then begin
           Priority := Priority - 30;
          end  {if}
          else begin
           Priority := 0
          end;  {else}
         end;  {if}
         NumRate := Priority + 1;   {to adjust to queue numbers}
         if Queue[NumRate].Full = False then begin
          QueueFullFalse(Queue,NumRate,QueueBegin, QueueEnd,
                  InfoPriority, InfoType, PullTest, Size,
                  ModelTime);
         end  {if}
         else if (Queue[NumRate].Full = True) then begin
```

```
      QueueFullTrue(NumRate,Queue,InfoPriority,InfoType,
                PullTest,Size,ModelTime);
    end; {else}
    QueueLength := QueueLength + 1;
  end; {else}
  Msg1Up := False;
  MsgGen1(ModelTime,Msg1Time);
end; {if}
if Msg2Up then begin
  {writeln('       Msg2Up  ',ModelTime:5:3);}
  InfoPriority := Random(4);
  InfoType := Random(14);
  Size2 := Random(2000);
  Size := Size2 + 1;
  UserPull := Random(4);
  if (UserPull = 3) then begin
    PushNumb := PushNumb + 1;
    PullTest := False;
  end
  else begin
    PullNumb := PullNumb + 1;
    PullTest := True;
  end;
  InfoTemp2 := InfoType + 1;  { to get queue numbering}
  PriTemp2 := InfoPriority + 1;  { to get queue numbering}
  CountMatrix[PriTemp2,InfoTemp2] :=
CountMatrix[PriTemp2,InfoTemp2] + 1;
```

```
if (QueueLength = 0) and (QueuePullTime <= ModelTime) and
(not QueuePullUp) then begin
 QueuePullUp := True;
 QueuePullTime := ModelTime;
 MsgUp := True;
end  {if}
else begin   {place in queue}
 Priority := InfoPriority + Size + InfoType;
 if PullTest then begin
  if Priority > 30 then begin
   Priority := Priority - 30;
  end  {if}
  else begin
   Priority := 0
  end;  {else}
 end;  {if}
 NumRate := Priority + 1;  {to adjust to queue numbers}
 if Queue[NumRate].Full = False then begin
  QueueFullFalse(Queue,NumRate,QueueBegin, QueueEnd,
         InfoPriority, InfoType, PullTest, Size,
         ModelTime);
 end  {if}
 else if (Queue[NumRate].Full = True) then begin
  QueueFullTrue(NumRate,Queue,InfoPriority,InfoType,
         PullTest,Size,ModelTime);
 end; {else if}
 QueueLength := QueueLength + 1;
```

```
    end;  {else}
    Msg2Up := False;
    MsgGen2(ModelTime,Msg2Time);
  end;  {if}
  if QueuePullUp and (QueueLength > 0) then begin
    {writeln('Queue Pull Up with QL > 0     ',ModelTime:5:3);}
    MsgSize := Queue[QueueBegin].Next^.FileSize;
    MsgTime := (MsgSize/DataRate);
    QueuePullTime := ModelTime + MsgTime;
    MsgSent := MsgSent + 1;
    InfoTemp := Queue[QueueBegin].Next^.FileInfoType;
    InfoTemp2 := InfoTemp + 1;   {to get queue numbering}
    PriTemp := Queue[QueueBegin].Next^.FileInfoPriority;
    PriTemp2 := PriTemp + 1;  {to queue numbering}
    MsgSentMatrix[PriTemp2,InfoTemp2] :=
MsgSentMatrix[PriTemp2,InfoTemp2]
     + 1;
    CountMatrix[PriTemp2,InfoTemp2] :=
CountMatrix[PriTemp2,InfoTemp2] - 1;
    MsgWaitTime := ModelTime -
Queue[QueueBegin].Next^.FileTimeEnter;
    if Queue[QueueBegin].Next^.FileUserPull then begin
      PullWaitTime := PullWaitTime + MsgWaitTime;
      PullSent := PullSent + 1;
      PullAveWait := PullWaitTime/PullSent;
    end
    else begin
      PushWaitTime := PushWaitTime + MsgWaitTime;
```

```
          PushSent := PushSent + 1;
          PushAveWait := PushWaitTime/PushSent;
         end;
         QueueWaitTime := QueueWaitTime + MsgWaitTime;
         MsgWaitMatrix[PriTemp2,InfoTemp2] :=
MsgWaitMatrix[PriTemp2,InfoTemp2]
          + MsgWaitTime;
         AveWait := QueueWaitTime/MsgSent;
         TempAve := MsgWaitMatrix[PriTemp2,InfoTemp2]/
         (MsgSentMatrix[PriTemp2,InfoTemp2]);
         MsgAveWaitMatrix[PriTemp2,InfoTemp2] := TempAve;
         QueueLength := QueueLength - 1;
         if QueueLength > 0 then begin
          if (Queue[QueueBegin].Next^.Next = nil) then begin
            Queue[QueueBegin].Full := False;
            dispose(Queue[QueueBegin].Next);
            Queue[QueueBegin].Next := nil;
            Temp := Queue[QueueBegin].NextNum;
            Queue[QueueBegin].NextNum := 0;
            QueueBegin := Temp;
            Queue[QueueBegin].PrevNum := 0;
          end {if}
          else if (Queue[QueueBegin].Next^.Next <> nil) then begin
            TempPtr := Queue[QueueBegin].Next;
            Queue[QueueBegin].Next := Queue[QueueBegin].Next^.Next;
            dispose(TempPtr);
            TempPtr := nil;
          end;   {else if}
```

```
end {if}
else if QueueLength = 0 then begin
 Queue[QueueBegin].Full := False;
 dispose(Queue[QueueBegin].Next);
 Queue[QueueBegin].Next := nil;
 Queue[QueueBegin].PrevNum := 0;
 Queue[QueueBegin].NextNum := 0;
 QueueBegin := 0;
 QueueEnd := maxint;
end; {else if}
QueuePullUp := False;
end {if}
else if QueuePullUp and MsgUp and (QueueLength = 0) then begin
 {writeln(' QueuePullUp and QL = 0 ',ModelTime:5:3);}
 MsgTime := Size/DataRate;
 QueuePullTime := ModelTime + MsgTime;
 MsgSent := MsgSent + 1;
 InfoTemp2 := InfoType + 1;
 PriTemp2 := InfoPriority + 1;
 AveWait := QueueWaitTime/MsgSent;
 if PullTest then begin
  PullSent := PullSent + 1;
  PullAveWait := PullWaitTime/PullSent;
 end
 else begin
  PushSent := PushSent + 1;
  PushAveWait := PushWaitTime/PushSent;
 end;
```

```
        MsgSentMatrix[PriTemp2,InfoTemp2] :=
MsgSentMatrix[PriTemp2,InfoTemp2]
         + 1;
        TempAve := MsgWaitMatrix[PriTemp2,InfoTemp2]/
         (MsgSentMatrix[PriTemp2,InfoTemp2]);
        MsgAveWaitMatrix[PriTemp2,InfoTemp2] := TempAve;
        CountMatrix[PriTemp2,InfoTemp2] :=
CountMatrix[PriTemp2,InfoTemp2] - 1;
        QueuePullUp := False;
        MsgUp := False;
       end   {else}
       else if QueuePullUp and not(MsgUp) and (QueueLength = 0) then
begin
        QueuePullUp := False;
       end;  {else}
       if QueueReDoUp and (QueueLength > 0) then begin
        {writeln('QueueRe Do Up  ',QueueLength,'   ',ModelTime:5:3);}
        QueuePtr := QueueBegin;
        while QueuePtr <> QueueEnd do begin
         TempInfoPri := Queue[QueuePtr].Next^.FileInfoPriority;
         TempInfoType := Queue[QueuePtr].Next^.FileInfoType;
         TempUsePull := Queue[QueuePtr].Next^.FileUserPull;
         TempSize := Queue[QueuePtr].Next^.FileSize;
         TempTime := Queue[QueuePtr].Next^.FileTimeEnter;
         HoldTime := ModelTime - TempTime;
         if Queue[QueuePtr].Next^.Next <> nil then begin
          TempPtr2 := Queue[QueuePtr].Next^.Next;
          dispose(Queue[QueuePtr].Next);
```

```
  Queue[QueuePtr].Next := nil;
  Queue[QueuePtr].Next := TempPtr2;
end
else if Queue[QueuePtr].Next^.Next = nil then begin
  dispose(Queue[QueuePtr].Next);
  Queue[QueuePtr].Next := nil;
  Temp := Queue[QueuePtr].NextNum;
  Temp2 := Queue[QueuePtr].PrevNum;
  Queue[Temp2].NextNum := Queue[QueuePtr].NextNum;
  Queue[Temp].PrevNum := Queue[QueuePtr].PrevNum;
  Queue[QueuePtr].PrevNum := 0;
  Queue[QueuePtr].NextNum := 0;
  Queue[QueuePtr].Full := False;
  if QueueBegin = QueuePtr then begin
    QueueBegin := Temp;
  end;
  QueuePtr := Temp;
end;   { else}
Priority := TempInfoPri + TempSize + TempInfoType;
if (TempUsePull = True) then begin
  if Priority > 30 then begin
    Priority := Priority - 30;
  end  {if}
  else begin
    Priority := 0
  end;  {else}
end;  {if}
if HoldTime <= 3.0 then begin
```

```
 if Priority > 5 then begin
  Priority := Priority - 5;
 end {if}
 else begin
  Priority := 0
 end; {else}
end {if}
else if (3.0 < HoldTime) and (HoldTime <= 9.0) then begin
 if Priority > 20 then begin
  Priority := Priority - 20;
 end { if}
 else begin
  Priority := 0
 end; {else}
end {else if}
else if (9.0 < HoldTime) and (HoldTime <= 15.0) then begin
 if Priority > 40 then begin
  Priority := Priority - 40;
 end { if}
 else begin
  Priority := 0
 end; {else}
end {else if}
else if 15.0 < HoldTime then begin
 if Priority > 60 then begin
  Priority := Priority - 60;
 end {if}
 else begin
```

```
      Priority := 0
     end;  {else}
    end;  {else if}
    NumRate := Priority + 1;   {to adjust to queue numbers}
    if NumRate = QueuePtr then begin
     NumRate := NumRate - 1;
    end;
    if Queue[NumRate].Full = False then begin
     QueueFullFalse(Queue,NumRate,QueueBegin, QueueEnd,
            TempInfoPri,TempInfoType,TempUsePull, TempSize,
            TempTime);
    end  {if}
    else if (Queue[NumRate].Full = True) then begin
     QueueFullTrue(NumRate,Queue,TempInfoPri,TempInfoType,
            TempUsePull,TempSize,TempTime);
    end;  {else if}
   end;  {while}
   QueueReDoUp := False;
   QueueReDoTimeGen(ModelTime,QueueReDoTime);
  end;  {if}
  if QueueReDoUp and (QueueLength = 0) then begin
   QueueReDoUp := False;
   QueueReDoTimeGen(ModelTime,QueueReDoTime);
  end;  {if}
  Counter := Counter + 1;
  {writeln('Counter equals ',Counter,'ModelTime is ',ModelTime);}
 end;  {while}
```

```
if QueueLength > 0 then begin
 Temp := QueueBegin;
 while Queue[Temp].NextNum <> 0 do begin
  while Queue[Temp].Next <> nil do begin
   Ptr := Queue[Temp].Next^.Next;
   dispose(Queue[Temp].Next);
   Queue[Temp].Next := Ptr;
   Ptr := nil;
  end; { while}
  Temp := Queue[Temp].NextNum;
 end; {while}
 while Queue[Temp].Next <> nil do begin
  Ptr := Queue[Temp].Next;
  Queue[Temp].Next := Queue[Temp].Next^.Next;
  dispose(Ptr);
  Ptr := nil;
 end; { while}
end; {if}
TotalMsg := QueueLength + MsgSent;
PercentSent := MsgSent/TotalMsg;
PercentQueue := QueueLength/TotalMsg;
write(outfile,'2,');
write(outfile,L,',');
write(outfile,DataRate,',');
write(outfile,ArrivalRate1,',');
write(outfile,ArrivalRate2,',');
Write(outfile,ModelTime:5:3,',');
Write(outfile,QueueLength,',');
```

```
Write(outfile,MsgSent,',');
Write(outfile,AveWait:5:3,',');
write(outfile,TotalMsg,',');
write(outfile,PercentSent:5:3,',');
write(outfile,PercentQueue:5:3,',');
write(outfile,PullNumb,',');
write(outfile,PullSent,',');
write(outfile,PullAveWait:5:3,',');
write(outfile,PushNumb,',');
write(outfile,PushSent,',');
write(outfile,PushAveWait:5:3,',');
write(outfile,MsgSentMatrix[1,1],',');
write(outfile,MsgAveWaitMatrix[1,1]:5:3,',');
write(outfile,CountMatrix[1,1],',');
write(outfile,MsgSentMatrix[1,2],',');
write(outfile,MsgAveWaitMatrix[1,2]:5:3,',');
write(outfile,CountMatrix[1,2],',');
write(outfile,MsgSentMatrix[1,3],',');
write(outfile,MsgAveWaitMatrix[1,3]:5:3,',');
write(outfile,CountMatrix[1,3],',');
write(outfile,MsgSentMatrix[1,4],',');
write(outfile,MsgAveWaitMatrix[1,4]:5:3,',');
write(outfile,CountMatrix[1,4],',');
write(outfile,MsgSentMatrix[1,5],',');
write(outfile,MsgAveWaitMatrix[1,5]:5:3,',');
write(outfile,CountMatrix[1,5],',');
write(outfile,MsgSentMatrix[1,6],',');
write(outfile,MsgAveWaitMatrix[1,6]:5:3,',');
```

```
write(outfile,CountMatrix[1,6],',');
write(outfile,MsgSentMatrix[1,7],',');
write(outfile,MsgAveWaitMatrix[1,7]:5:3,',');
write(outfile,CountMatrix[1,7],',');
write(outfile,MsgSentMatrix[1,8],',');
write(outfile,MsgAveWaitMatrix[1,8]:5:3,',');
write(outfile,CountMatrix[1,8],',');
write(outfile,MsgSentMatrix[1,9],',');
write(outfile,MsgAveWaitMatrix[1,9]:5:3,',');
write(outfile,CountMatrix[1,9],',');
write(outfile,MsgSentMatrix[1,10],',');
write(outfile,MsgAveWaitMatrix[1,10]:5:3,',');
write(outfile,CountMatrix[1,10],',');
write(outfile,MsgSentMatrix[1,11],',');
write(outfile,MsgAveWaitMatrix[1,11]:5:3,',');
write(outfile,CountMatrix[1,11],',');
write(outfile,MsgSentMatrix[1,12],',');
write(outfile,MsgAveWaitMatrix[1,12]:5:3,',');
write(outfile,CountMatrix[1,12],',');
write(outfile,MsgSentMatrix[1,13],',');
write(outfile,MsgAveWaitMatrix[1,13]:5:3,',');
write(outfile,CountMatrix[1,13],',');
write(outfile,MsgSentMatrix[1,14],',');
write(outfile,MsgAveWaitMatrix[1,14]:5:3,',');
write(outfile,CountMatrix[1,14],',');
write(outfile,MsgSentMatrix[2,1],',');
write(outfile,MsgAveWaitMatrix[2,1]:5:3,',');
write(outfile,CountMatrix[2,1],',');
```

```
write(outfile,MsgSentMatrix[2,2],',');
write(outfile,MsgAveWaitMatrix[2,2]:5:3,',');
write(outfile,CountMatrix[2,2],',');
write(outfile,MsgSentMatrix[2,3],',');
write(outfile,MsgAveWaitMatrix[2,3]:5:3,',');
write(outfile,CountMatrix[2,3],',');
write(outfile,MsgSentMatrix[2,4],',');
write(outfile,MsgAveWaitMatrix[2,4]:5:3,',');
write(outfile,CountMatrix[2,4],',');
write(outfile,MsgSentMatrix[2,5],',');
write(outfile,MsgAveWaitMatrix[2,5]:5:3,',');
write(outfile,CountMatrix[2,5],',');
write(outfile,MsgSentMatrix[2,6],',');
write(outfile,MsgAveWaitMatrix[2,6]:5:3,',');
write(outfile,CountMatrix[2,6],',');
write(outfile,MsgSentMatrix[2,7],',');
write(outfile,MsgAveWaitMatrix[2,7]:5:3,',');
write(outfile,CountMatrix[2,7],',');
write(outfile,MsgSentMatrix[2,8],',');
write(outfile,MsgAveWaitMatrix[2,8]:5:3,',');
write(outfile,CountMatrix[2,8],',');
write(outfile,MsgSentMatrix[2,9],',');
write(outfile,MsgAveWaitMatrix[2,9]:5:3,',');
write(outfile,CountMatrix[2,9],',');
write(outfile,MsgSentMatrix[2,10],',');
write(outfile,MsgAveWaitMatrix[2,10]:5:3,',');
write(outfile,CountMatrix[2,10],',');
write(outfile,MsgSentMatrix[2,11],',');
```

```
write(outfile,MsgAveWaitMatrix[2,11]:5:3,',');
write(outfile,CountMatrix[2,11],',');
write(outfile,MsgSentMatrix[2,12],',');
write(outfile,MsgAveWaitMatrix[2,12]:5:3,',');
write(outfile,CountMatrix[2,12],',');
write(outfile,MsgSentMatrix[2,13],',');
write(outfile,MsgAveWaitMatrix[2,13]:5:3,',');
write(outfile,CountMatrix[2,13],',');
write(outfile,MsgSentMatrix[2,14],',');
write(outfile,MsgAveWaitMatrix[2,14]:5:3,',');
write(outfile,CountMatrix[2,14],',');
write(outfile,MsgSentMatrix[3,1],',');
write(outfile,MsgAveWaitMatrix[3,1]:5:3,',');
write(outfile,CountMatrix[3,1],',');
write(outfile,MsgSentMatrix[3,2],',');
write(outfile,MsgAveWaitMatrix[3,2]:5:3,',');
write(outfile,CountMatrix[3,2],',');
write(outfile,MsgSentMatrix[3,3],',');
write(outfile,MsgAveWaitMatrix[3,3]:5:3,',');
write(outfile,CountMatrix[3,3],',');
write(outfile,MsgSentMatrix[3,4],',');
write(outfile,MsgAveWaitMatrix[3,4]:5:3,',');
write(outfile,CountMatrix[3,4],',');
write(outfile,MsgSentMatrix[3,5],',');
write(outfile,MsgAveWaitMatrix[3,5]:5:3,',');
write(outfile,CountMatrix[3,5],',');
write(outfile,MsgSentMatrix[3,6],',');
write(outfile,MsgAveWaitMatrix[3,6]:5:3,',');
```

```
write(outfile,CountMatrix[3,6],',');
write(outfile,MsgSentMatrix[3,7],',');
write(outfile,MsgAveWaitMatrix[3,7]:5:3,',');
write(outfile,CountMatrix[3,7],',');
write(outfile,MsgSentMatrix[3,8],',');
write(outfile,MsgAveWaitMatrix[3,8]:5:3,',');
write(outfile,CountMatrix[3,8],',');
write(outfile,MsgSentMatrix[3,9],',');
write(outfile,MsgAveWaitMatrix[3,9]:5:3,',');
write(outfile,CountMatrix[3,9],',');
write(outfile,MsgSentMatrix[3,10],',');
write(outfile,MsgAveWaitMatrix[3,10]:5:3,',');
write(outfile,CountMatrix[3,10],',');
write(outfile,MsgSentMatrix[3,11],',');
write(outfile,MsgAveWaitMatrix[3,11]:5:3,',');
write(outfile,CountMatrix[3,11],',');
write(outfile,MsgSentMatrix[3,12],',');
write(outfile,MsgAveWaitMatrix[3,12]:5:3,',');
write(outfile,CountMatrix[3,12],',');
write(outfile,MsgSentMatrix[3,13],',');
write(outfile,MsgAveWaitMatrix[3,13]:5:3,',');
write(outfile,CountMatrix[3,13],',');
write(outfile,MsgSentMatrix[3,14],',');
write(outfile,MsgAveWaitMatrix[3,14]:5:3,',');
write(outfile,CountMatrix[3,14],',');
write(outfile,MsgSentMatrix[4,1],',');
write(outfile,MsgAveWaitMatrix[4,1]:5:3,',');
write(outfile,CountMatrix[4,1],',');
```

```
write(outfile,MsgSentMatrix[4,2],',');
write(outfile,MsgAveWaitMatrix[4,2]:5:3,',');
write(outfile,CountMatrix[4,2],',');
write(outfile,MsgSentMatrix[4,3],',');
write(outfile,MsgAveWaitMatrix[4,3]:5:3,',');
write(outfile,CountMatrix[4,3],',');
write(outfile,MsgSentMatrix[4,4],',');
write(outfile,MsgAveWaitMatrix[4,4]:5:3,',');
write(outfile,CountMatrix[4,4],',');
write(outfile,MsgSentMatrix[4,5],',');
write(outfile,MsgAveWaitMatrix[4,5]:5:3,',');
write(outfile,CountMatrix[4,5],',');
write(outfile,MsgSentMatrix[4,6],',');
write(outfile,MsgAveWaitMatrix[4,6]:5:3,',');
write(outfile,CountMatrix[4,6],',');
write(outfile,MsgSentMatrix[4,7],',');
write(outfile,MsgAveWaitMatrix[4,7]:5:3,',');
write(outfile,CountMatrix[4,7],',');
write(outfile,MsgSentMatrix[4,8],',');
write(outfile,MsgAveWaitMatrix[4,8]:5:3,',');
write(outfile,CountMatrix[4,8],',');
write(outfile,MsgSentMatrix[4,9],',');
write(outfile,MsgAveWaitMatrix[4,9]:5:3,',');
write(outfile,CountMatrix[4,9],',');
write(outfile,MsgSentMatrix[4,10],',');
write(outfile,MsgAveWaitMatrix[4,10]:5:3,',');
write(outfile,CountMatrix[4,10],',');
write(outfile,MsgSentMatrix[4,11],',');
```

```
  write(outfile,MsgAveWaitMatrix[4,11]:5:3,',');
  write(outfile,CountMatrix[4,11],',');
  write(outfile,MsgSentMatrix[4,12],',');
  write(outfile,MsgAveWaitMatrix[4,12]:5:3,',');
  write(outfile,CountMatrix[4,12],',');
  write(outfile,MsgSentMatrix[4,13],',');
  write(outfile,MsgAveWaitMatrix[4,13]:5:3,',');
  write(outfile,CountMatrix[4,13],',');
  write(outfile,MsgSentMatrix[4,14],',');
  write(outfile,MsgAveWaitMatrix[4,14]:5:3,',');
  write(outfile,CountMatrix[4,14],',');
  write(outfile,MsgSentMatrix[4,15],',');
  write(outfile,MsgAveWaitMatrix[4,15]:5:3,',');
  writeln(outfile,CountMatrix[4,15],',');
end;  {Prioritize}
end.  {Thesis2c}
```

APPENDIX C: DATA RESULTS

A. RUN ONE

```
MTB > Retrieve  'FINAL10.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: FINAL10.WK1
No matching ranges; using default conversion.
MTB > AOVOneway 'I' 'V'.

ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS         F        p
FACTOR      1  0.137344  0.137344    525.37    0.000
ERROR      98  0.025620  0.000261
TOTAL      99  0.162964
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
LEVEL       N      MEAN     STDEV  ------+---------+---------+---------+
I          50   0.21586   0.02059                          (*-)
V          50   0.14174   0.00995   (-*-)
                                   ------+---------+---------+---------+
POOLED STDEV =  0.01617               0.150     0.175     0.200     0.225
MTB > #pull v pull
MTB > AOVOneway 'O' 'AB'.

ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS         F        p
FACTOR      1  0.156658  0.156658    531.94    0.000
ERROR      98  0.028861  0.000295
TOTAL      99  0.185519
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
LEVEL       N      MEAN     STDEV  --------+---------+---------+--------
O          50   0.21468   0.02214                          (-*-)
AB         50   0.13552   0.00993  (-*-)
                                   --------+---------+---------+--------
POOLED STDEV =  0.01716                 0.150     0.175     0.200
MTB > #push v push
MTB > AOVOneway c18 c31.
```

```
ANALYSIS OF VARIANCE
SOURCE    DF        SS        MS        F       p
FACTOR     1  0.119094  0.119094   449.20   0.000
ERROR     98  0.025982  0.000265
TOTAL     99  0.145077
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
LEVEL     N      MEAN     STDEV  ---+---------+---------+---------+---
R              50   0.21708  0.02009                          (-*-)
AE        50  0.14806  0.01126   (-*-)
                                 ---+---------+---------+---------+---
POOLED STDEV =  0.01628           0.150     0.175     0.200     0.225
MTB > #pull v push alg1
MTB > AOVOneway c15 c18.

ANALYSIS OF VARIANCE
SOURCE    DF        SS        MS        F       p
FACTOR     1  0.000144  0.000144     0.32   0.572
ERROR     98  0.043803  0.000447
TOTAL     99  0.043947
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
LEVEL     N      MEAN     STDEV  ---------+---------+---------+-------
O         50  0.21468   0.02214  (--------------*--------------)
R         50  0.21708   0.02009       (--------------*--------------)
                                 ---------+---------+---------+-------
POOLED STDEV =  0.02114               0.2120    0.2160    0.2200
MTB > #pull v push alg2
MTB > AOVOneway c28 c31 .

ANALYSIS OF VARIANCE
SOURCE    DF        SS        MS        F       p
FACTOR     1  0.003931  0.003931    34.89   0.000
ERROR     98  0.011041  0.000113
TOTAL     99  0.014973
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
LEVEL     N      MEAN     STDEV  -----+---------+---------+---------+-
AB        50  0.13552   0.00993  (-----*-----)
AE        50  0.14806   0.01126                       (-----*-----)
                                 -----+---------+---------+---------+-
```

```
POOLED STDEV = 0.01061            0.1350    0.1400    0.1450    0.1500
MTB > nooutfile
```

B. RUN TWO

```
MTB > Retrieve  'FINAL11.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: FINAL11.WK1
No matching ranges; using default conversion.
MTB > #ave alg1 v ave alg2
MTB > AOVOneway c9 c22 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS       MS        F        p
FACTOR     1  1.30531  1.30531   159.07    0.000
ERROR     98  0.80417  0.00821
TOTAL     99  2.10948
                                     INDIVIDUAL 95 PCT CI'S FOR MEAN
                                     BASED ON POOLED STDEV
 LEVEL     N     MEAN     STDEV  ---+---------+---------+---------+---
I         50    0.55480  0.11511                             (--*---)
V         50    0.32630  0.05622   (--*--)
                                 ---+---------+---------+---------+---
POOLED STDEV = 0.09059            0.320     0.400     0.480     0.560
MTB > #pull v pull
MTB > AOVOneway c15 c28 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS       MS        F        p
FACTOR     1  1.78009  1.78009   224.73    0.000
ERROR     98  0.77626  0.00792
TOTAL     99  2.55635
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
 LEVEL     N     MEAN     STDEV  ----+---------+---------+---------+--
O         50   0.55638   0.11710                        (--*-)
AB        50   0.28954   0.04614(--*-)
                                 ----+---------+---------+---------+--
```

```
POOLED STDEV =  0.08900          0.30     0.40     0.50     0.60
MTB > #push v push
MTB > AOVOneway c18 c31 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS        F        p
FACTOR     1  0.90573   0.90573   100.59   0.000
ERROR     98  0.88238   0.00900
TOTAL     99  1.78812
                                INDIVIDUAL 95 PCT CI'S FOR MEAN
                                BASED ON POOLED STDEV
LEVEL      N     MEAN     STDEV --+---------+---------+---------+----
R         50   0.55314   0.11493                           (---*---)
AE        50   0.36280   0.06928 (---*---)
                                --+---------+---------+---------+----
POOLED STDEV =  0.09489         0.350     0.420     0.490     0.560
MTB > #pull v push alg1
MTB > AOVOneway c15 c18 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS       MS       F       p
FACTOR     1   0.0003   0.0003    0.02   0.889
ERROR     98   1.3192   0.0135
TOTAL     99   1.3195
                                INDIVIDUAL 95 PCT CI'S FOR MEAN
                                BASED ON POOLED STDEV
LEVEL      N     MEAN     STDEV ----------+---------+---------+------
O         50   0.5564    0.1171  (---------------*---------------)
R         50   0.5531    0.1149 (----------------*---------------)
                                ----------+---------+---------+------
POOLED STDEV =  0.1160               0.540     0.560     0.580
MTB > #pull v push alg2
MTB > AOVOneway c28 c31 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS        F       p
FACTOR     1  0.13418   0.13418   38.74   0.000
ERROR     98  0.33945   0.00346
TOTAL     99  0.47363
                                INDIVIDUAL 95 PCT CI'S FOR MEAN
                                BASED ON POOLED STDEV
LEVEL      N     MEAN     STDEV ---------+---------+---------+-------
```

```
AB          50    0.28954   0.04614  (-----*----)
AE          50    0.36280   0.06928                          (-----*----)
                                      ---------+---------+---------+-------
POOLED STDEV =  0.05885                    0.300     0.330     0.360
MTB > nooutfile
```

C. RUN THREE

```
MTB > Retrieve  'FINAL12.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: FINAL12.WK1
No matching ranges; using default conversion.
MTB > #ave wait alg1 v ave wait alg2
MTB > AOVOneway c9 c22 .

ANALYSIS OF VARIANCE
SOURCE     DF       SS        MS        F        p
FACTOR      1   7.6602    7.6602   112.99    0.000
ERROR      98   6.6437    0.0678
TOTAL      99  14.3039
                                 INDIVIDUAL 95 PCT CI'S FOR MEAN
                                 BASED ON POOLED STDEV
 LEVEL      N     MEAN     STDEV  --+---------+---------+---------+----
I          50    1.1879    0.3025                          (--*---)
V          50    0.6344    0.2099  (---*--)
                                  --+---------+---------+---------+----
POOLED STDEV =  0.2604          0.60      0.80      1.00      1.20
MTB > #pull v pull
MTB > AOVOneway c15 c28 .

ANALYSIS OF VARIANCE
SOURCE     DF       SS        MS        F        p
FACTOR      1  11.1356   11.1356   203.05    0.000
ERROR      98   5.3744    0.0548
TOTAL      99  16.5100
                                  INDIVIDUAL 95 PCT CI'S FOR MEAN
                                  BASED ON POOLED STDEV
 LEVEL      N     MEAN     STDEV  --+---------+---------+---------+----
O          50    1.1913    0.3044                        (--*-)
```

```
AB          50    0.5239    0.1306  (--*--)
                                     --+---------+---------+---------+----
POOLED STDEV =   0.2342              0.50      0.75      1.00      1.25
MTB > #push v push
MTB > AOVOneway c18 c31 .

ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS        F        p
FACTOR      1    4.8316    4.8316    53.64    0.000
ERROR      98    8.8275    0.0901
TOTAL      99   13.6591
                                     INDIVIDUAL 95 PCT CI'S FOR MEAN
                                     BASED ON POOLED STDEV
 LEVEL      N      MEAN     STDEV  -------+---------+---------+---------
R          50    1.1845    0.3028                      (---*---)
AE         50    0.7449    0.2975  (---*---)
                                    -------+---------+---------+---------
POOLED STDEV =   0.3001                 0.80      1.00      1.20
MTB > #pull v push alg1
MTB > AOVOneway c15 c18 .

ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS        F        p
FACTOR      1    0.0012    0.0012     0.01    0.911
ERROR      98    9.0308    0.0922
TOTAL      99    9.0319
                                     INDIVIDUAL 95 PCT CI'S FOR MEAN
                                     BASED ON POOLED STDEV
 LEVEL      N      MEAN     STDEV  -+---------+---------+---------+-----
O          50    1.1913    0.3044    (----------------*----------------)
R          50    1.1845    0.3028  (----------------*----------------)
                                    -+---------+---------+---------+-----
POOLED STDEV =   0.3036             1.100     1.150     1.200     1.250
MTB > #pull v push alg2
MTB > AOVOneway c28 c31 .


ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS        F        p
FACTOR      1    1.2208    1.2208    23.14    0.000
ERROR      98    5.1711    0.0528
```

```
TOTAL      99   6.3919
                                       INDIVIDUAL 95 PCT CI'S FOR MEAN
                                       BASED ON POOLED STDEV
LEVEL      N     MEAN     STDEV  -----+---------+---------+---------+-
AB         50    0.5239   0.1306   (-----*------)
AE         50    0.7449   0.2975                      (-----*------)
                                 -----+---------+---------+---------+-
POOLED STDEV =  0.2297              0.50      0.60      0.70      0.80
MTB > nooutfile
```

D. RUN FOUR

```
MTB > Retrieve  'FINAL13.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: FINAL13.WK1
No matching ranges; using default conversion.
MTB > #ave wait alg1 v ave wait alg2
MTB > AOVOneway c9 c22 .

ANALYSIS OF VARIANCE
SOURCE     DF       SS         MS         F        p
FACTOR      1 0.0042380 0.0042380  2993.55    0.000
ERROR      98 0.0001387 0.0000014
TOTAL      99 0.0043768
                                      INDIVIDUAL 95 PCT CI'S FOR MEAN
                                      BASED ON POOLED STDEV
LEVEL      N     MEAN     STDEV  -------+---------+---------+---------
I          50    0.038660  0.001479                              (*
V          50    0.025640  0.000802(*)
                                 -------+---------+---------+---------
POOLED STDEV = 0.001190              0.0280    0.0320    0.0360
MTB > #pull v pull
MTB > AOVOneway c15 c28 .


ANALYSIS OF VARIANCE
SOURCE     DF       SS         MS         F         p
FACTOR      1 0.0054023 0.0054023  3320.08    0.000
ERROR      98 0.0001595 0.0000016
```

```
TOTAL      99 0.0055617
                                        INDIVIDUAL 95 PCT CI'S FOR MEAN
                                        BASED ON POOLED STDEV
LEVEL      N     MEAN      STDEV  ---+---------+---------+---------+---
O          50    0.038580  0.001579                               (*)
AB         50    0.023880  0.000872(*
                                  ---+---------+---------+---------+---
POOLED STDEV = 0.001276          0.0250    0.0300    0.0350    0.0400
MTB > #push v push
MTB > AOVOneway c18 c31 .

ANALYSIS OF VARIANCE
SOURCE     DF       SS         MS          F          p
FACTOR      1 0.0037700 0.0037700   2355.02    0.000
ERROR      98 0.0001569 0.0000016
TOTAL      99 0.0039268
                                        INDIVIDUAL 95 PCT CI'S FOR MEAN
                                        BASED ON POOLED STDEV
LEVEL      N     MEAN      STDEV  -----+---------+---------+---------+-
R          50    0.038680  0.001558                               (*)
AE         50    0.026400  0.000881(*)
                                  -----+---------+---------+---------+-
POOLED STDEV = 0.001265          0.0280    0.0320    0.0360    0.0400
MTB > #pull v push alg1
MTB > AOVOneway c15 c18 .

ANALYSIS OF VARIANCE
SOURCE     DF       SS         MS          F       p
FACTOR      1 0.0000002 0.0000002     0.10    0.751
ERROR      98 0.0002411 0.0000025
TOTAL      99 0.0002413
                                        INDIVIDUAL 95 PCT CI'S FOR MEAN
                                        BASED ON POOLED STDEV
LEVEL      N     MEAN      STDEV  ---------+---------+---------+-------
O          50    0.038580  0.001579 (--------------*--------------)
R          50    0.038680  0.001558    (-------------*--------------)
                                  ---------+---------+---------+-------
POOLED STDEV = 0.001568           0.03840   0.03870   0.03900
MTB > #pull v push alg2
MTB > AOVOneway c28 c31.

ANALYSIS OF VARIANCE
```

```
SOURCE    DF       SS        MS         F        p
FACTOR     1 0.0001588 0.0001588   206.67    0.000
ERROR     98 0.0000753 0.0000008
TOTAL     99 0.0002340
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
LEVEL      N     MEAN     STDEV  ----+---------+---------+---------+--
AB        50   0.023880  0.000872 --*-)
AE        50   0.026400  0.000881                    (-*-)
                                 ----+---------+---------+---------+--
POOLED STDEV = 0.000876         0.0240    0.0250    0.0260    0.0270
MTB > nooutfile
```

E. RUN FIVE

```
MTB > Retrieve  'FINAL14.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: FINAL14.WK1
No matching ranges; using default conversion.
MTB > #ave wait alg1 v ave wait alg2
MTB > AOVOneway c9 c22 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS         F        p
FACTOR     1 0.0379470 0.0379470  1513.58    0.000
ERROR     98 0.0024570 0.0000251
TOTAL     99 0.0404040
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
LEVEL      N     MEAN     STDEV  --+---------+---------+---------+----
I         50   0.098080  0.006439                               (*)
V         50   0.059120  0.002946 (*)
                                 --+---------+---------+---------+----
POOLED STDEV = 0.005007          0.060     0.072     0.084     0.096

MTB > #pull v pull
MTB > AOVOneway c15 c28 .

ANALYSIS OF VARIANCE
```

```
SOURCE     DF       SS        MS         F         p
FACTOR      1 0.0566916 0.0566916  2371.73    0.000
ERROR      98 0.0023425 0.0000239
TOTAL      99 0.0590341
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
LEVEL      N     MEAN     STDEV   --------+---------+---------+--------
O         50   0.097980  0.006454                               (*)
AB        50   0.050360  0.002481(*
                                  --------+---------+---------+--------
POOLED STDEV = 0.004889               0.060     0.075     0.090
MTB > #push v push
MTB > AOVOneway c18 c31 .

ANALYSIS OF VARIANCE
SOURCE     DF       SS        MS         F         p
FACTOR      1 0.0311876 0.0311876  1176.85    0.000
ERROR      98 0.0025971 0.0000265
TOTAL      99 0.0337846
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
LEVEL      N     MEAN     STDEV   ---------+---------+---------+-------
R         50   0.098100  0.006447                                (*)
AE        50   0.062780  0.003382(*-)
                                  ---------+---------+---------+-------
POOLED STDEV = 0.005148               0.072     0.084     0.096
MTB > #pull v push alg1
MTB > AOVOneway c15 c18 .

ANALYSIS OF VARIANCE
SOURCE     DF       SS        MS         F       p
FACTOR      1 0.0000004 0.0000004   0.01    0.926
ERROR      98 0.0040775 0.0000416
TOTAL      99 0.0040778
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV


LEVEL      N     MEAN     STDEV   ---------+---------+---------+-------
O         50   0.097980  0.006454(-------------*--------------)
R         50   0.098100  0.006447 (-------------*--------------)
                                  ---------+---------+---------+-------
```

```
POOLED STDEV = 0.006450              0.0972    0.0984    0.0996
MTB > #pull v push alg2
MTB > AOVOneway c28 c31 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS         F        p
FACTOR     1 0.0038564 0.0038564  438.38    0.000
ERROR     98 0.0008621 0.0000088
TOTAL     99 0.0047185
                                 INDIVIDUAL 95 PCT CI'S FOR MEAN
                                 BASED ON POOLED STDEV
 LEVEL    N     MEAN     STDEV  -------+---------+---------+---------
AB       50   0.050360 0.002481 (-*-)
AE       50   0.062780 0.003382                          (-*-)
                                -------+---------+---------+---------
POOLED STDEV = 0.002966              0.0520    0.0560    0.0600
MTB > nooutfile
```

F. RUN SIX

```
MTB > Retrieve 'FINAL15.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: FINAL15.WK1
No matching ranges; using default conversion.
MTB > #ave wait alg1 v ave wait alg2
MTB > AOVOneway c9 c22 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS        F        p
FACTOR     1  0.235128  0.235128   353.73   0.000
ERROR     98  0.065142  0.000665
TOTAL     99  0.300270
                                 INDIVIDUAL 95 PCT CI'S FOR MEAN
                                 BASED ON POOLED STDEV

 LEVEL    N     MEAN     STDEV  --------+---------+---------+--------
I        50   0.21940  0.03228                            (-*-)
V        50   0.12242  0.01695  (-*-)
                                --------+---------+---------+--------
```

```
POOLED STDEV =  0.02578              0.140    0.175    0.210
MTB > #pull v pull
MTB > AOVOneway c15 c28 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS       MS        F        p
FACTOR     1  0.409856 0.409856  723.71    0.000
ERROR     98  0.055500 0.000566
TOTAL     99  0.465356
                                 INDIVIDUAL 95 PCT CI'S FOR MEAN
                                 BASED ON POOLED STDEV
LEVEL      N     MEAN    STDEV   ---------+---------+---------+-------
O         50   0.21902  0.03210                              (-*)
AB        50   0.09098  0.01011 (-*)
                                 ---------+---------+---------+-------
POOLED STDEV =  0.02380              0.120    0.160    0.200
MTB > #push v push
MTB > AOVOneway c18 c31 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS       MS        F        p
FACTOR     1  0.175980 0.175980  240.45    0.000
ERROR     98  0.071723 0.000732
TOTAL     99  0.247703
                                 INDIVIDUAL 95 PCT CI'S FOR MEAN
                                 BASED ON POOLED STDEV
LEVEL      N     MEAN    STDEV   --------+---------+---------+--------
R         50   0.21958  0.03248                            (-*--)
AE        50   0.13568  0.02021  (-*--)
                                 --------+---------+---------+--------
POOLED STDEV =  0.02705              0.150    0.180    0.210
MTB > #pull v push alg1
MTB > AOVOneway c15 c18 .
```

```
ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS       F       p
FACTOR     1  0.00001   0.00001    0.01   0.931
ERROR     98  0.10220   0.00104
TOTAL     99  0.10221
                                INDIVIDUAL 95 PCT CI'S FOR MEAN
                                BASED ON POOLED STDEV
 LEVEL    N     MEAN    STDEV  -+---------+---------+---------+-----
O        50   0.21902  0.03210  (--------------*--------------)
R        50   0.21958  0.03248   (--------------*--------------)
                               -+---------+---------+---------+-----
POOLED STDEV =  0.03229      .2100    0.2160    0.2220    0.2280
MTB > #pull v push alg2
MTB > AOVOneway c28 c31 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS         MS        F         p
FACTOR     1  0.049952  0.049952  195.61    0.000
ERROR     98  0.025026  0.000255
TOTAL     99  0.074978
                                INDIVIDUAL 95 PCT CI'S FOR MEAN
                                BASED ON POOLED STDEV
 LEVEL    N     MEAN    STDEV  ---+---------+---------+---------+---
AB       50   0.09098  0.01011  (--*--)
AE       50   0.13568  0.02021                              (--*--)
                               ---+---------+---------+---------+---
POOLED STDEV =  0.01598         0.090     0.105     0.120     0.135
MTB > nooutfile
```

G. RUN SEVEN

```
MTB > Retrieve  'FINAL16.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: FINAL16.WK1
No matching ranges; using default conversion.
MTB > #ave wait alg1 v ave wait alg2
MTB > AOVOneway c9 c22 .
```

```
ANALYSIS OF VARIANCE
SOURCE     DF        SS          MS          F          p
FACTOR      1 0.0143520 0.0143520  1656.42    0.000
ERROR      98 0.0008491 0.0000087
TOTAL      99 0.0152012
                                         INDIVIDUAL 95 PCT CI'S FOR MEAN
                                         BASED ON POOLED STDEV
LEVEL      N     MEAN       STDEV   ----+---------+---------+---------+--
I          50    0.071200  0.003586                                (*)
V          50    0.047240  0.002115 (*-)
                                    ----+---------+---------+---------+--
POOLED STDEV = 0.002944             0.0490    0.0560    0.0630    0.0700
MTB > #pull v pull
MTB > AOVOneway c15 c28 .

ANALYSIS OF VARIANCE
SOURCE     DF        SS          MS          F          p
FACTOR      1 0.0182250 0.0182250  1934.13    0.000
ERROR      98 0.0009234 0.0000094
TOTAL      99 0.0191484
                                         INDIVIDUAL 95 PCT CI'S FOR MEAN
                                         BASED ON POOLED STDEV
LEVEL      N     MEAN       STDEV   ------+---------+---------+---------+
O          50    0.071160  0.003825                                (*)
AB         50    0.044160  0.002054 (*)
                                    ------+---------+---------+---------+
POOLED STDEV = 0.003070             0.0480    0.0560    0.0640    0.0720
MTB > #push v push
MTB > AOVOneway c18 c31 .

ANALYSIS OF VARIANCE
SOURCE     DF        SS          MS          F          p
FACTOR      1 0.0125216 0.0125216  1320.70    0.000
ERROR      98 0.0009291 0.0000095
TOTAL      99 0.0134507
                                         INDIVIDUAL 95 PCT CI'S FOR MEAN
                                         BASED ON POOLED STDEV
LEVEL      N     MEAN       STDEV   --+---------+---------+---------+----
R          50    0.071140  0.003603                               (-*)
AE         50    0.048760  0.002446(-*)
                                    --+---------+---------+---------+----
POOLED STDEV = 0.003079             0.0490    0.0560    0.0630    0.0700
```

```
MTB > #pull v push alg1
MTB > AOVOneway c15 c18 .

ANALYSIS OF VARIANCE
SOURCE    DF      SS         MS         F       p
FACTOR     1 0.0000000 0.0000000    0.00    0.979
ERROR     98 0.0013527 0.0000138
TOTAL     99 0.0013527
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
 LEVEL     N    MEAN    STDEV  --+---------+---------+---------+----
O         50   0.071160 0.003825  (---------------*----------------)
R         50   0.071140 0.003603 (----------------*----------------)
                               --+---------+---------+---------+----
POOLED STDEV = 0.003715       7020   0.07080   0.07140   0.07200
MTB > #pull v push alg2
MTB > AOVOneway c28 c31 .

ANALYSIS OF VARIANCE
SOURCE    DF      SS         MS         F       p
FACTOR     1 0.0005290 0.0005290  103.72   0.000
ERROR     98 0.0004998 0.0000051
TOTAL     99 0.0010288
                                   INDIVIDUAL 95 PCT CI'S FOR MEAN
                                   BASED ON POOLED STDEV
 LEVEL     N    MEAN    STDEV  --------+---------+---------+--------
AB        50   0.044160 0.002054 (---*---)
AE        50   0.048760 0.002446                     (---*---)
                               --------+---------+---------+--------
POOLED STDEV = 0.002258         0.0448    0.0464    0.0480
MTB > nooutfile
```

H. RUN EIGHT

```
MTB > Retrieve  'FINAL17.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: FINAL17.WK1
No matching ranges; using default conversion.
MTB > #ave wait alg1 v ave wait alg2
```

```
MTB > AOVOneway c9 c22 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS        F        p
FACTOR     1  0.139428  0.139428  1014.58   0.000
ERROR     98  0.013468  0.000137
TOTAL     99  0.152895
                              INDIVIDUAL 95 PCT CI'S FOR MEAN
                              BASED ON POOLED STDEV
LEVEL     N     MEAN     STDEV  ---------+---------+---------+-------
I        50   0.18212  0.01252                               (*)
V        50   0.10744  0.01087(*)
                                ---------+---------+---------+-------
POOLED STDEV =  0.01172              0.125     0.150     0.175
MTB > #pull v pull
MTB > AOVOneway c15 c28 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS        F        p
FACTOR     1  0.202140  0.202140  1721.86   0.000
ERROR     98  0.011505  0.000117
TOTAL     99  0.213645
                              INDIVIDUAL 95 PCT CI'S FOR MEAN
                              BASED ON POOLED STDEV
LEVEL     N     MEAN     STDEV  -+---------+---------+---------+-----
O        50   0.18232  0.01349                                 (*)
AB       50   0.09240  0.00727   (*)
                                -+---------+---------+---------+-----
POOLED STDEV =  0.01083        0.090     0.120     0.150     0.180
MTB > #push v push
MTB > AOVOneway c18 c31  .

ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS        F        p
FACTOR     1  0.112225  0.112225   697.86   0.000
ERROR     98  0.015760  0.000161
TOTAL     99  0.127985
                              INDIVIDUAL 95 PCT CI'S FOR MEAN
                              BASED ON POOLED STDEV
```

```
 LEVEL      N     MEAN    STDEV  ------+---------+---------+---------+
R          50   0.18194  0.01226                          (-*)
AE         50   0.11494  0.01309 (*)
                                 ------+---------+---------+---------+
POOLED STDEV =  0.01268           0.125     0.150     0.175     0.200
MTB > #pull v push alg1
MTB > AOVOneway c15 c18 .

ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS        F        p
FACTOR      1  0.000004  0.000004     0.02    0.883
ERROR      98  0.016280  0.000166
TOTAL      99  0.016283
                                 INDIVIDUAL 95 PCT CI'S FOR MEAN
                                 BASED ON POOLED STDEV
 LEVEL      N     MEAN    STDEV  -------+---------+---------+---------
O          50   0.18232  0.01349  (------------*-------------)
R          50   0.18194  0.01226 (-------------*------------)
                                 -------+---------+---------+---------
POOLED STDEV =  0.01289          0.1800    0.1825    0.1850
MTB > #pull v push alg2
MTB > AOVOneway c28 c31 .

ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS        F        p
FACTOR      1  0.012701  0.012701   113.31    0.000
ERROR      98  0.010985  0.000112
TOTAL      99  0.023686
                                 INDIVIDUAL 95 PCT CI'S FOR MEAN
                                 BASED ON POOLED STDEV
 LEVEL      N     MEAN    STDEV  ---------+---------+---------+-------
AB         50   0.09240  0.00727  (--*---)
AE         50   0.11494  0.01309                           (---*--)
                                 ---------+---------+---------+-------
POOLED STDEV =  0.01059             0.0960    0.1040    0.1120
MTB > nooutfile
```

I. RUN NINE

```
MTB > Retrieve  'FINAL18.WK1';
```

```
SUBC>  Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: FINAL18.WK1
No matching ranges; using default conversion.
MTB > #ave wait alg1 v ave wait alg2
MTB > AOVOneway c9 c22 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS        F       p
FACTOR     1  0.97654   0.97654   233.44   0.000
ERROR     98  0.40996   0.00418
TOTAL     99  1.38650
                                 INDIVIDUAL 95 PCT CI'S FOR MEAN
                                 BASED ON POOLED STDEV
LEVEL      N    MEAN     STDEV  --+---------+---------+---------+----
I         50  0.41586  0.08341                              (-*--)
V         50  0.21822  0.03755   (-*--)
                                --+---------+---------+---------+----
POOLED STDEV =  0.06468         0.210     0.280     0.350     0.420
MTB > #pull v pull
MTB > AOVOneway c15 c28 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS        F       p
FACTOR     1  1.54331   1.54331   395.95   0.000
ERROR     98  0.38198   0.00390
TOTAL     99  1.92528
                                 INDIVIDUAL 95 PCT CI'S FOR MEAN
                                 BASED ON POOLED STDEV
LEVEL      N    MEAN     STDEV  --+---------+---------+---------+----
O         50  0.41618  0.08447                                (-*-)
AB        50  0.16772  0.02570   (-*-)
                                --+---------+---------+---------+----
POOLED STDEV =  0.06243         0.160     0.240     0.320     0.400
MTB > #push v push
MTB > AOVOneway c18 c31 .

ANALYSIS OF VARIANCE
SOURCE    DF       SS        MS        F       p
FACTOR     1  0.74184   0.74184   166.21   0.000
ERROR     98  0.43739   0.00446
TOTAL     99  1.17922
```

```
                                      INDIVIDUAL 95 PCT CI'S FOR MEAN
                                      BASED ON POOLED STDEV
 LEVEL     N      MEAN     STDEV  ---+---------+---------+---------+---
 R        50   0.41584   0.08333                             (--*--)
 AE       50   0.24358   0.04453  (---*--)
                                  ---+---------+---------+---------+---
POOLED STDEV =  0.06681             0.240     0.300     0.360     0.420
MTB > #pull v push alg1
MTB > AOVOneway c15 c18 .

ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS        F        p
FACTOR      1   0.00000   0.00000     0.00    0.984
ERROR      98   0.68982   0.00704
TOTAL      99   0.68983
                                      INDIVIDUAL 95 PCT CI'S FOR MEAN
                                      BASED ON POOLED STDEV
 LEVEL     N      MEAN     STDEV  ---------+---------+---------+-------
 O        50   0.41618   0.08447  (--------------*---------------)
 R        50   0.41584   0.08333  (--------------*---------------)
                                  ---------+---------+---------+-------
POOLED STDEV =  0.08390                0.405     0.420     0.435
MTB > #pull v push alg2
MTB > AOVOneway c28 c31 .

ANALYSIS OF VARIANCE
SOURCE     DF        SS        MS        F        p
FACTOR      1   0.14387   0.14387   108.84    0.000
ERROR      98   0.12954   0.00132
TOTAL      99   0.27341
                                      INDIVIDUAL 95 PCT CI'S FOR MEAN
                                      BASED ON POOLED STDEV
 LEVEL     N      MEAN     STDEV  --------+---------+---------+--------
 AB       50   0.16772   0.02570  (--*--)
 AE       50   0.24358   0.04453                           (--*---)
                                  --------+---------+---------+--------
POOLED STDEV =  0.03636               0.180     0.210     0.240
MTB > nooutfile
```

J. FLASH PRECEDENCE COMPARISON

```
RUN ONE
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N      MEAN     STDEV  SE MEAN
C3  50  0.06971  0.00218  0.00031
C6  50   0.3236   0.0564   0.0080

95 PCT CI FOR MU C3 - MU C6: (-0.26990, -0.2378)

TTEST MU C3 = MU C6 (VS NE): T= -31.79  P=0.0000  DF=  49

MTB > nooutfile

RUN TWO
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N      MEAN     STDEV  SE MEAN
C3  50  0.05348  0.00138  0.00019
C6  50   0.1387   0.0113   0.0016

95 PCT CI FOR MU C3 - MU C6: (-0.08845, -0.0820)

TTEST MU C3 = MU C6 (VS NE): T= -53.03  P=0.0000  DF=  50

MTB > nooutfile

RUN THREE
MTB > Retrieve  '12FLASH.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 12FLASH.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '12FLASH.WK1';
```

```
SUBC>  Lotus.
12FLASH.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 12FLASH.WK1
MTB > TwoSample 95.0 C3 C6;
SUBC>  Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50  0.07918  0.00229  0.00032
C6  50    0.634    0.223    0.032

95 PCT CI FOR MU C3 - MU C6: (-0.61825, -0.491)

TTEST MU C3 = MU C6 (VS NE): T= -17.60  P=0.0000  DF=  49

MTB > nooutfile

RUN FOUR
MTB > Retrieve  '13FLASH.WK1';
SUBC>  Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 13FLASH.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > TwoSample 95.0 C3 C6;
SUBC>  Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50 0.009755 0.000142 0.000020
C6  50 0.025551 0.000903  0.00013

95 PCT CI FOR MU C3 - MU C6: (-0.016056, -0.01554)

TTEST MU C3 = MU C6 (VS NE): T= -122.21  P=0.0000  DF=  51

MTB > Save  '13FLASH.WK1';
SUBC>  Lotus.
13FLASH.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
```

```
  LOTUS 1-2-3 file: 13FLASH.WK1
MTB > nooutfile

RUN FIVE
MTB > Retrieve  '14FLASH.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 14FLASH.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN      STDEV   SE MEAN
C3  50  0.012698  0.000171  0.000024
C6  50   0.05893   0.00360   0.00051

95 PCT CI FOR MU C3 - MU C6: (-0.047259, -0.04521)

TTEST MU C3 = MU C6 (VS NE): T= -90.78  P=0.0000  DF=  49

MTB > Save  '14FLASH.WK1';
SUBC>   Lotus.
14FLASH.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 14FLASH.WK1
MTB > nooutfile

RUN SIX
MTB > Save  '15FLASH.WK1';
SUBC>   Lotus.
15FLASH.WK1 already exists.
MTB > Retrieve  '15FLASH.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 15FLASH.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > TwoSample 95.0 C3 C6;
```

```
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50  0.014321  0.000201  0.000028
C6  50   0.1200    0.0165    0.0023

95 PCT CI FOR MU C3 - MU C6: (-0.110393, -0.1010)

TTEST MU C3 = MU C6 (VS NE): T= -45.38  P=0.0000  DF=  49

MTB > nooutfile

RUN SEVEN
MTB > Retrieve  '16FLASH.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 16FLASH.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50  0.017832  0.000264  0.000037
C6  50   0.04686   0.00233   0.00033

95 PCT CI FOR MU C3 - MU C6: (-0.029693, -0.02836)

TTEST MU C3 = MU C6 (VS NE): T= -87.47  P=0.0000  DF=  50

MTB > nooutfile

RUN EIGHT
MTB > Retrieve  '17FLASH.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 17FLASH.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
```

```
MTB > let c6 = c4/c5
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV  SE MEAN
C3  50  0.023252  0.000357  0.000051
C6  50    0.1055    0.0120    0.0017

95 PCT CI FOR MU C3 - MU C6: (-0.085699, -0.0789)

TTEST MU C3 = MU C6 (VS NE): T= -48.60  P=0.0000  DF=  49

MTB > nooutfile

RUN NINE
MTB > Retrieve  '18FLASH.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 18FLASH.WK1
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > TwoSample 95.0 'C' 'F';
SUBC>   Alternative 0.

TWOSAMPLE T FOR C VS F
    N     MEAN     STDEV  SE MEAN
C  50  0.023252  0.000357  0.000051
F  50    0.1055    0.0120    0.0017

95 PCT CI FOR MU C - MU F: (-0.085699, -0.0789)

TTEST MU C = MU F (VS NE): T= -48.60  P=0.0000  DF=  49

MTB > nooutfile
```

K. IMMEDIATE PRECEDENCE COMPARISON

```
RUN ONE
MTB > Retrieve  '10IMM.WK1';
```

```
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 10IMM.WK1
No matching ranges; using default conversion.
MTB > let c3 =c1/c2
MTB > let c6 = c4/c5
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN    STDEV   SE MEAN
C3  50  0.08853  0.00359  0.00051
C6  50   0.1394  0.0110   0.0016

95 PCT CI FOR MU C3 - MU C6: (-0.05414, -0.0476)

TTEST MU C3 = MU C6 (VS NE): T= -31.16  P=0.0000  DF=  59

MTB > Save  '10IMM.WK1';
SUBC>   Lotus.
10IMM.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 10IMM.WK1
MTB > nooutfile

RUN TWO
MTB > Retrieve  '11IMM.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 11IMM.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '11IMM.WK1';
SUBC>   Lotus.
11IMM.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 11IMM.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
```

```
    N    MEAN     STDEV   SE MEAN
C3  50  0.12734  0.00591  0.00084
C6  50   0.3225   0.0573   0.0081

95 PCT CI FOR MU C3 - MU C6: (-0.21151, -0.1788)

TTEST MU C3 = MU C6 (VS NE): T= -23.97  P=0.0000  DF=  50

MTB > nooutfile

RUN THREE
MTB > Retrieve  '12IMM.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 12IMM.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '12IMM.WK1';
SUBC>   Lotus.
12IMM.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 12IMM.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N    MEAN     STDEV   SE MEAN
C3  50  0.15057  0.00562  0.00079
C6  50    0.635    0.228    0.032

95 PCT CI FOR MU C3 - MU C6: (-0.54965, -0.420)

TTEST MU C3 = MU C6 (VS NE): T= -15.01  P=0.0000  DF=  49

MTB > nooutfile

RUN FOUR
MTB > Retrieve  '13IMM.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 13IMM.WK1
```

```
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '13IMM.WK1';
SUBC>   Lotus.
13IMM.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 13IMM.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
     N    MEAN      STDEV     SE MEAN
C3  50  0.016152  0.000282  0.000040
C6  50  0.025661  0.000873   0.00012

95 PCT CI FOR MU C3 - MU C6: (-0.009769, -0.00925)

TTEST MU C3 = MU C6 (VS NE): T= -73.33  P=0.0000  DF=  59

MTB > nooutfile

RUN FIVE
MTB > Retrieve  '14IMM.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 14IMM.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '14IMM.WK1';
SUBC>   Lotus.
14IMM.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 14IMM.WK1
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
     N    MEAN      STDEV     SE MEAN
C3  50  0.023166  0.000533  0.000075
C6  50   0.05900   0.00364   0.00051
```

95 PCT CI FOR MU C3 - MU C6: (-0.036876, -0.03479)

TTEST MU C3 = MU C6 (VS NE): T= -68.84 P=0.0000 DF= 51

MTB > nooutfile

RUN SIX
MTB > Retrieve '15IMM.WK1';
SUBC> Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
LOTUS 1-2-3 file: 15IMM.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save '15IMM.WK1';
SUBC> Lotus.
15IMM.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
LOTUS 1-2-3 file: 15IMM.WK1
MTB > TwoSample 95.0 C3 C6;
SUBC> Alternative 0.

TWOSAMPLE T FOR C3 VS C6

	N	MEAN	STDEV	SE MEAN
C3	50	0.027286	0.000586	0.000083
C6	50	0.1220	0.0180	0.0025

95 PCT CI FOR MU C3 - MU C6: (-0.099882, -0.0896)

TTEST MU C3 = MU C6 (VS NE): T= -37.18 P=0.0000 DF= 49

MTB > nooutfile

RUN SEVEN
MTB > Retrieve '16IMM.WK1';
SUBC> Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
LOTUS 1-2-3 file: 16IMM.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5

```
MTB > Save '16IMM.WK1';
SUBC>   Lotus.
16IMM.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 16IMM.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N      MEAN     STDEV   SE MEAN
C3  50  0.029553  0.000643  0.000091
C6  50   0.04716   0.00254   0.00036

95 PCT CI FOR MU C3 - MU C6: (-0.018348, -0.01686)

TTEST MU C3 = MU C6 (VS NE): T= -47.52  P=0.0000  DF=  55

MTB > nooutfile

RUN EIGHT
MTB > Retrieve '17IMM.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 17IMM.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save '17IMM.WK1';
SUBC>   Lotus.
17IMM.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 17IMM.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N      MEAN     STDEV   SE MEAN
C3  50   0.04239   0.00105   0.00015
C6  50    0.1074    0.0122    0.0017

95 PCT CI FOR MU C3 - MU C6: (-0.06849, -0.0615)
```

```
TTEST MU C3 = MU C6 (VS NE): T= -37.59  P=0.0000  DF=  49

MTB > nooutfile

RUN NINE
MTB > Retrieve  '18IMM.WK1';
SUBC>  Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 18IMM.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '18IMM.WK1';
SUBC>  Lotus.
18IMM.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 18IMM.WK1
MTB > TwoSample 95.0 C3 C6;
SUBC>  Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N    MEAN     STDEV   SE MEAN
C3  50  0.05044  0.00124  0.00018
C6  50   0.2155   0.0408   0.0058

95 PCT CI FOR MU C3 - MU C6: (-0.17662, -0.1534)

TTEST MU C3 = MU C6 (VS NE): T= -28.60  P=0.0000  DF=  49

MTB > nooutfile
```

L. PRIORITY PRECEDENCE COMPARISON

```
RUN ONE
MTB > Retrieve  '10PRIOR.WK1';
SUBC>  Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 10PRIOR.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
```

```
MTB > let c6 = c4/c5
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50   0.1799   0.0119   0.0017
C6  50   0.1450   0.0125   0.0018

95 PCT CI FOR MU C3 - MU C6: (0.0300, 0.0397)

TTEST MU C3 = MU C6 (VS NE): T= 14.29  P=0.0000  DF=  97

MTB > Save  '10PRIOR.WK1';
SUBC>   Lotus.
10PRIOR.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 10PRIOR.WK1
MTB > nooutfile

RUN TWO
MTB > Retrieve  '11PRIOR.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 11PRIOR.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '11PRIOR.WK1';
SUBC>   Lotus.
11PRIOR.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 11PRIOR.WK1
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50   0.3048   0.0247   0.0035
C6  50   0.3277   0.0674   0.0095

95 PCT CI FOR MU C3 - MU C6: (-0.0432, -0.0026)
```

```
TTEST MU C3 = MU C6 (VS NE): T= -2.26  P=0.028  DF= 61

MTB > nooutfile

RUN THREE
MTB > Retrieve  '12PRIOR.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 12PRIOR.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '12PRIOR.WK1';
SUBC>   Lotus.
12PRIOR.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 12PRIOR.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50   0.3989   0.0346   0.0049
C6  50    0.623    0.188    0.027

95 PCT CI FOR MU C3 - MU C6: (-0.2787, -0.170)

TTEST MU C3 = MU C6 (VS NE): T= -8.29  P=0.0000  DF= 52

MTB > nooutfile

RUN FOUR
MTB > Retrieve  '13PRIOR.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 13PRIOR.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '13PRIOR.WK1';
SUBC>   Lotus.
```

```
13PRIOR.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 13PRIOR.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN      STDEV    SE MEAN
C3  50  0.032311  0.000905   0.00013
C6  50  0.025639  0.000970   0.00014

95 PCT CI FOR MU C3 - MU C6: (0.00630, 0.00704)

TTEST MU C3 = MU C6 (VS NE): T= 35.56  P=0.0000  DF=  97

MTB > nooutfile

RUN FIVE
MTB > Retrieve  '14PRIOR.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 14PRIOR.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '14PRIOR.WK1';
SUBC>   Lotus.
14PRIOR.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 14PRIOR.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN      STDEV    SE MEAN
C3  50   0.05506   0.00176   0.00025
C6  50   0.05877   0.00346   0.00049

95 PCT CI FOR MU C3 - MU C6: (-0.00481, -0.00262)

TTEST MU C3 = MU C6 (VS NE): T= -6.76  P=0.0000  DF=  72
```

```
MTB > nooutfile

RUN SIX
MTB > Retrieve  '15PRIOR.WK1';
SUBC>  Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 15PRIOR.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '15PRIOR.WK1';
SUBC>  Lotus.
15PRIOR.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 15PRIOR.WK1
MTB > TwoSample 95.0 C3 C6;
SUBC>  Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N    MEAN     STDEV  SE MEAN
C3  50  0.07215  0.00249  0.00035
C6  50   0.1222   0.0180   0.0025

95 PCT CI FOR MU C3 - MU C6: (-0.05518, -0.0449)

TTEST MU C3 = MU C6 (VS NE): T= -19.50  P=0.0000  DF=  50

MTB > nooutfile

RUN SEVEN
MTB > Retrieve  '16PRIOR.WK1';
SUBC>  Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 16PRIOR.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '16PRIOR.WK1';
SUBC>  Lotus.
16PRIOR.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 16PRIOR.WK1
```

```
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50  0.05925  0.00182  0.00026
C6  50  0.04708  0.00238  0.00034

95 PCT CI FOR MU C3 - MU C6: (0.01133, 0.01302)

TTEST MU C3 = MU C6 (VS NE): T= 28.75  P=0.0000  DF=  91

MTB > nooutfile

RUN EIGHT
MTB > Retrieve  '17PRIOR.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 17PRIOR.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '17PRIOR.WK1';
SUBC>   Lotus.
17PRIOR.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 17PRIOR.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50  0.10084  0.00395  0.00056
C6  50   0.1077   0.0110   0.0016

95 PCT CI FOR MU C3 - MU C6: (-0.01013, -0.0035)

TTEST MU C3 = MU C6 (VS NE): T= -4.15  P=0.0001  DF=  61

MTB > nooutfile
```

```
RUN NINE
MTB > Save  '18PRIOR.WK1';
SUBC>   Lotus.
18PRIOR.WK1 already exists.
MTB > Retrieve  '18PRIOR.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 18PRIOR.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '18PRIOR.WK1';
SUBC>   Lotus.
18PRIOR.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 18PRIOR.WK1
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N    MEAN     STDEV    SE MEAN
C3  50  0.13333  0.00567  0.00080
C6  50   0.2199   0.0373   0.0053

95 PCT CI FOR MU C3 - MU C6: (-0.09731, -0.0759)

TTEST MU C3 = MU C6 (VS NE): T= -16.22  P=0.0000  DF=  51

MTB > nooutfile
```

M. ROUTINE PRECEDENCE COMPARISON

```
RUN ONE
MTB > Retrieve  '10ROUT.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 10ROUT.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
```

```
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50   0.5438   0.0743    0.011
C6  50   0.1440   0.0133   0.0019

95 PCT CI FOR MU C3 - MU C6: (0.378, 0.4212)

TTEST MU C3 = MU C6 (VS NE): T= 37.46  P=0.0000  DF=  52

MTB > Save  '10ROUT.WK1';
SUBC>   Lotus.
10ROUT.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 10ROUT.WK1
MTB > nooutfile

RUN TWO
MTB > Retrieve  '11ROUT.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 11ROUT.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '11ROUT.WK1';
SUBC>   Lotus.
11ROUT.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 11ROUT.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50    1.725    0.453    0.064
C6  50   0.3314   0.0634   0.0090

95 PCT CI FOR MU C3 - MU C6: (1.264, 1.5233)
```

TTEST MU C3 = MU C6 (VS NE): T= 21.56 P=0.0000 DF= 50

MTB > nooutfile

RUN THREE
MTB > Retrieve '12ROUT.WK1';
SUBC> Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
LOTUS 1-2-3 file: 12ROUT.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save '12ROUT.WK1';
SUBC> Lotus.
12ROUT.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
LOTUS 1-2-3 file: 12ROUT.WK1
MTB > TwoSample 95.0 C3 C6;
SUBC> Alternative 0.

TWOSAMPLE T FOR C3 VS C6

	N	MEAN	STDEV	SE MEAN
C3	50	4.14	1.19	0.17
C6	50	0.645	0.236	0.033

95 PCT CI FOR MU C3 - MU C6: (3.15, 3.838)

TTEST MU C3 = MU C6 (VS NE): T= 20.30 P=0.0000 DF= 52

MTB > nooutfile

RUN FOUR
MTB > Retrieve '13ROUT.WK1';
SUBC> Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
LOTUS 1-2-3 file: 13ROUT.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save '13ROUT.WK1';
SUBC> Lotus.
13ROUT.WK1 already exists.

```
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 13ROUT.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
     N     MEAN     STDEV   SE MEAN
C3  50  0.09641  0.00512  0.00072
C6  50  0.02584  0.00114  0.00016

95 PCT CI FOR MU C3 - MU C6: (0.06908, 0.07206)

TTEST MU C3 = MU C6 (VS NE): T= 95.05  P=0.0000  DF=  53

MTB > nooutfile

RUN FIVE
MTB > Retrieve  '14ROUT.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 14ROUT.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '14ROUT.WK1';
SUBC>   Lotus.
14ROUT.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 14ROUT.WK1
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
     N     MEAN     STDEV   SE MEAN
C3  50   0.3015    0.0243    0.0034
C6  50  0.05978  0.00333  0.00047

95 PCT CI FOR MU C3 - MU C6: (0.2348, 0.24868)

TTEST MU C3 = MU C6 (VS NE): T= 69.77  P=0.0000  DF=  50

MTB > nooutfile
```

RUN SIX

```
MTB > Retrieve '15ROUT.WK1';
SUBC> Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 15ROUT.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save '15ROUT.WK1';
SUBC> Lotus.
15ROUT.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 15ROUT.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC> Alternative 0.

TWOSAMPLE T FOR C3 VS C6
     N    MEAN     STDEV   SE MEAN
C3  50   0.764     0.127    0.018
C6  50   0.1255    0.0182   0.0026

95 PCT CI FOR MU C3 - MU C6: (0.602, 0.6747)

TTEST MU C3 = MU C6 (VS NE): T= 35.21  P=0.0000  DF=  51

MTB > nooutfile
```

RUN SEVEN

```
MTB > Retrieve '16ROUT.WK1';
SUBC> Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 16ROUT.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save '16ROUT.WK1';
SUBC> Lotus.
16ROUT.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 16ROUT.WK1
MTB > TwoSample 95.0 c3 c6;
SUBC> Alternative 0.
```

```
TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50   0.1777   0.0126   0.0018
C6  50  0.04769  0.00252  0.00036

95 PCT CI FOR MU C3 - MU C6: (0.1263, 0.13361)

TTEST MU C3 = MU C6 (VS NE): T= 71.64  P=0.0000  DF=  52

MTB > nooutfile

RUN EIGHT
MTB > Retrieve  '17ROUT.WK1';
SUBC>   Lotus.
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 17ROUT.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '17ROUT.WK1';
SUBC>   Lotus.
17ROUT.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 17ROUT.WK1
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50   0.5619   0.0469   0.0066
C6  50   0.1091   0.0128   0.0018

95 PCT CI FOR MU C3 - MU C6: (0.4390, 0.4666)

TTEST MU C3 = MU C6 (VS NE): T= 65.81  P=0.0000  DF=  56

MTB > nooutfile

RUN NINE
MTB > Retrieve  '18ROUT.WK1';
SUBC>   Lotus.
```

```
Converting Lotus 1-2-3 v2/v3 to MINITAB
  LOTUS 1-2-3 file: 18ROUT.WK1
No matching ranges; using default conversion.
MTB > let c3 = c1/c2
MTB > let c6 = c4/c5
MTB > Save  '18ROUT.WK1';
SUBC>   Lotus.
18ROUT.WK1 already exists.
Converting MINITAB to Lotus 1-2-3 version 2 or 3
  LOTUS 1-2-3 file: 18ROUT.WK1
MTB > TwoSample 95.0 C3 C6;
SUBC>   Alternative 0.

TWOSAMPLE T FOR C3 VS C6
    N     MEAN     STDEV   SE MEAN
C3  50    1.456    0.333    0.047
C6  50   0.2233   0.0408   0.0058

95 PCT CI FOR MU C3 - MU C6: (1.137, 1.3276)

TTEST MU C3 = MU C6 (VS NE): T= 25.98  P=0.0000  DF=  50

MTB > nooutfile
```

LIST OF REFERENCES

Groff, G.K., and Muth, J.F., *Operations Management*, Richard D. Irwin, Inc., 1972.

Rand, A.L., *The Communication Support System (CSS) and Its Planning and Management upon Implementation*, Master's Thesis, Naval Postgraduate School, Monterey, CA, March 1992.

Sproase, J., "User Pull-White Paper", Naval Command, Control and Ocean Surveillance Command (NCCOSC) Corporate Initiatives Group working paper, 16 November 1993.

U.S. Department of the Navy, Director, Space and Electronic Warfare, *The Copernicus Architecture*, Government Printing Office, Washington, DC, 1991.

U.S. Department of the Navy, Director, Space-Command and Control-Information Warfare, *Copernicus...Forward, C4I For The 21st Century*, Government Printing Office, Washington, DC, 1995.

U.S. Department of the Navy, Commander, Space and Naval Warfare Systems Command, *Joint Maritime Communications System Overview*, Commander, Space and Naval Warfare Systems Command, Washington, DC, 1997.

BIBLIOGRAPHY

Burns, P.J., *I Hate EXCEL 5*, Que Corporation, 1993.

Cooper, D., *Oh! Pascal!*, 3rd ed., W.W. Norton and Company, 1992.

MINITAB Reference Manual, Minitab, Inc, 1991.

Ryan, B.F., Joiner, B.L., and Ryan, T.A., *MINITAB Handbook*, 2nd ed., Duxbury Press, 1992.

Walpole, R.E., and Myers, R.H., *Probability and Statistics for Engineers and Scientists*, 4th ed., Macmillan Publishing Company, 1989.

Weiss, N.A., and Hassett, M.J., *Introductory Statistics*, 3rd ed., Addison-Wesley Publishing Company, 1991.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Professor Michael G. Sovereign Code CC/SV 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Professor Orrin E. Marvel Code CC/MA 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

5 Lieutenant Commander Christopher H. Halton 1
   Commander
   Destroyer Squadron 20
   FPO AE 09506-4719